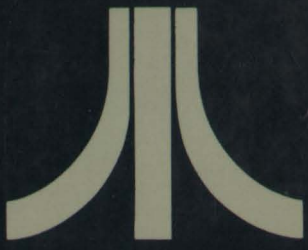
Versionen 600XL und 800XL enthalten

MEIN ATARI® COMPUTER

Der Schlüssel zum ATARI®-Privatcomputer



Lon Poole, Martin McNiff & Steven Cook

te-wi

MEIN ATARI® COMPUTER

Der Schlüssel zum ATARI®-Privatcomputer

Lon Poole, Martin McNiff & Steven Cook

te-wi

IMPRESSUM

Grundlage dieses Buches ist die Übersetzung der amerikanischen Originalausgabe "YOUR ATARI® COMPUTER" by Lon Poole with Martin McNiff & Steven Cook.

English edition Copyright © 1982
by McGraw-Hill, Inc.
All right reserved.

Übersetzung Copyright © 1983
by te-wi Verlag GmbH

Alle Rechte vorbehalten. Ohne ausdrückliche schriftliche Genehmigung der Herausgeber ist es nicht gestattet, das Buch oder Teile daraus in irgendeiner Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

LIZENZNEHMER und HERAUSGEBER:
te-wi Verlag GmbH, Theo-Prosel-Weg 1
8000 München 40

Der Herausgeber übernimmt keine Gewähr dafür, daß die beschriebenen Schaltungen, Baugruppen, Verfahren, Programme usw. funktionsfähig und frei von Schutzrechten Dritter sind.

Dieses Buch entstand nicht mit Unterstützung oder ausdrücklicher Billigung durch Atari, Inc. und stellt keine Veröffentlichung von Atari, Inc. dar. Alle im Buchtext benutzten Warenzeichen (TM) oder eingetragenen Warenzeichen ® sind Warenzeichen der Atari Inc.:

ATARI® ⅄ ®
ATARI® 400™ Computer
ATARI® 410™ Programmaufzeichnung
ATARI® 600™ Computer
ATARI® 800™ Computer
ATARI® 810™ Laufwerk
ATARI® 820™ 40zeiliger Drucker
ATARI® 822™ Thermischer Drucker
ATARI® 825™ 80zeiliger Drucker
ATARI® 830™ Akustisches Modem
ATARI® 850™ Interface-Modul
Star Raiders™
Music Composer™
Memory Module™

GESAMTHERSTELLUNG:
technik marketing, München
tm 3378/484 4. Auflage
Printed in Austria
ISBN 3-921803-18-7

Inhalt

	Vorwort	VII
	Einführung	VIII
1	Aufbau des ATARI-Computers	1-1
2	Bedienung des ATARI-Computers	2-1
3	Programmieren in BASIC	3-1
4	BASIC für Fortgeschrittene	4-1
5	Der Programmrecorder	5-1
6	Die ATARI-Drucker	6-1
7	Die ATARI-Diskettenstation 810	7-1
8	Einführung in die Grafikfunktionen des ATARI-Computers	8-1
9	Weiterführende Beschreibung der Grafikfunktionen des ATARI-Computers	9-1
10	Tonerzeugung mit dem ATARI-Computer	10-1
11	Zusammenfassung aller BASIC-Befehle und -Funktionen	11-1
A	Fehlermeldungen und Erklärungen dazu	A-1
B	Die Statuskodes	B-1
C	Abgeleitete Trigonometrische Funktionen	C-1
D	Kodes, Zeichen und die Funktion der Tastatur	D-1
E	Die BASIC-Befehle und -Funktionen und die dazugehörigen Abkürzungen	E-1
F	Die Speicheraufteilung	F-1
G	Einige oft benutzte Speicheradressen und deren Funktionen	G-1
H	Tabellen zur Zahlenumwandlung	H-1
I	Literaturhinweise	I-1
J	Neue ATARI-Produkte	J-1

VORWORT

Ohne die Unterstützung von ATARI wäre es nicht möglich gewesen, dieses Buch zu schreiben. Unser besonderer Dank gilt J. Peter Nelson und Sandy Bertino, die uns großzügigerweise eine Palette von Geräten zur Verfügung stellten. Dieselben Geräte wurden auch für die in diesem Buch enthaltenen Photographien verwendet. Damit war es uns möglich, alle Eigenschaften des ATARI-BASIC genau zu erkunden und zu beschreiben. Außerdem möchten wir an dieser Stelle Go Sugiura von AMDEK für die Überlassung eines Farbmonitors danken. In der Tat liefert ein Farbmonitor ein wesentlich schärferes Bild als ein Fernsehgerät.

Cynthia Greever hat die meisten der in diesem Buch enthaltenen Programme getestet und am Anhang mitgearbeitet. Schließlich möchten wir auch John Crane und seinen Kollegen von John Crane Consulting dafür danken, daß sie das Manuskript für dieses Buch überarbeitet und in vielen Punkten verbessert haben. Selbstverständlich übernehmen wir die Verantwortung für eventuell noch vorhandene Fehler oder Fehlinterpretationen.

EINFÜHRUNG

In diesem Buch werden der ATARI 400 und 800 sowie die gebräuchlichsten Peripheriegeräte, wie zum Beispiel die Diskettenstation oder die verschiedenen Drucker, beschrieben. Es wird davon ausgegangen, daß die einzelnen Komponenten Ihres Computersystemes entsprechend den dazugehörigen Beschreibungen untereinander und mit der Spannungsversorgung verbunden sind. Gegenstand dieser Beschreibung soll nicht sein, wie das System aufgebaut wird, sondern wie es benutzt wird.

Dieses Buch kann in drei große Teile unterteilt werden. Der erste Teil ist für den Anwender gedacht, der fertige Programme benutzen will und nur wenig oder gar keine Lust hat, selber Programme zu erstellen. Der zweite Teil wurde für den (zukünftigen) Programmierer geschrieben. Hier wird der Umgang mit dem BASIC* des ATARI-Computers beschrieben. Bei dem dritten Teil handelt es sich um eine weiter- und tiefergehende Beschreibung des ATARI-Computers. Dieser Teil ist für den Anwender gedacht, der die Grundlagen verstanden hat und nun das System näher kennenlernen möchte. Diese drei Teile können nicht als völlig voneinander getrennt betrachtet werden. Jemand, der den ersten Teil benutzt, wird sicher auch einmal im zweiten Teil nachsehen, wie bei der BASIC-Programmierung überhaupt vorgegangen wird. Genauso werden auch Anwender, die hauptsächlich mit dem zweiten und dritten Teil arbeiten, von Zeit zu Zeit einmal etwas im ersten Abschnitt nachschlagen müssen.

Die ersten beiden Kapitel sollen die Fragen beantworten, wie der ATARI-Computer funktioniert und wie man ihn bedient. Ein komplettes Computersystem besteht aus mehreren Komponenten, die untereinander verbunden sind. Im ersten Kapitel werden die einzelnen Komponenten und ihre Funktion beschrieben. Im zweiten Kapitel erfahren Sie, wie diese Komponenten bedient werden. Mit Hilfe dieser beiden Kapitel können Sie jedes käuflich zu erwerbende Programm (zum Beispiel für Textverarbeitung, Buchhaltung, Datenverwaltung, Unterricht oder Unterhaltung) benutzen.

Die Kapitel 3 bis 10 enthalten Informationen darüber, wie Sie selbst BASIC-Programme erstellen können. Die Grundlagen des ATARI-BASIC sind in Kapitel 3 beschrieben. Diese Beschreibung wird im vierten Kapitel fortgesetzt, wobei hier allerdings bereits etwas tiefer gegangen wird.

Einige besonders wichtige Punkte werden in eigenen Kapiteln ausführlich beschrieben. In Kapitel 5 erfahren Sie, wie Sie mit Hilfe des Programmrecorders Daten und Programme speichern und später wieder laden können. In Kapitel 6 wird beschrieben, wie die ATARI-Drucker angesprochen werden. Hierbei wird besonders auf den 80-Zeichen-Drucker ATARI 825 eingegangen. In Kapitel 7 wird erklärt, wie mit Hilfe der Diskettenstation Programme und Daten gespeichert und geladen werden können. In den Kapiteln 8 und 9 wird beschrieben, wie Grafiken auf dem Bildschirm erzeugt werden können. Hier finden sie außerdem verschiedene Möglichkeiten, wie das BASIC ausgeschaltet werden kann, wenn besondere grafische Effekte erzeugt werden sollen. In Kapitel 10 wird erklärt, wie mit Hilfe des ATARI-Computers Töne und Geräusche erzeugt werden können.

Danach beginnt der Nachschlageteil dieses Buches. In Kapitel 11 finden Sie eine ausführliche Beschreibung für jeden BASIC-Befehl und jede BASIC-Funktion, einschließlich der Befehle, die im Zusammenhang mit der Diskettenstation benutzt werden. Den Abschluß des Nachschlageteils bildet der Anhang.

** In diesem Buch wird nur das normale ATARI-BASIC, auch Sheperdson-BASIC genannt, beschrieben. Eine andere BASIC-Version, Microsoft-BASIC, kann über ATARI bezogen werden. Eine dritte Version wird unter dem Namen BASIC A+ von der Firma Optimized Systems Software (Cupertino, California) vertrieben. Es wird allerdings weder Microsoft-BASIC noch BASIC A+ in diesem Buch beschrieben.*

1

AUFBAU DES ATARI-COMPUTERS

Ein komplettes ATARI-Computersystem besteht aus mehreren Einzelgeräten. Ein solches System, dessen Kernstück der ATARI 800 ist, ist in Bild 1-1 abgebildet. Ihr eigenes System muß nicht unbedingt genauso aussehen, denn es gibt schließlich eine ziemlich große Menge von Zusatzgeräten. Allerdings müssen in jedem System die folgenden drei Komponenten enthalten sein: der ATARI 400 oder 800 selbst, die in den Computer integrierte Tastatur und ein Fernsehgerät. Diese Komponenten und einige gebräuchliche Zusatzgeräte sollen im folgenden näher betrachtet werden. In diesem Kapitel wird nicht beschrieben, wie die einzelnen Geräte untereinander und mit der Spannungsversorgung zu verbinden sind, denn dafür stehen Ihnen ja die entsprechenden Betriebsanleitungen zur Verfügung.

Die verschiedenen Komponenten eines Computersystemes

Den ATARI-Computer gibt es in zwei verschiedenen Ausführungen. Der ATARI 400 (Bild 1-2) und 800 (Bild 1-3) sind allerdings elektrisch vollkommen identisch. Sie unterscheiden sich nur im mechanischen Aufbau. Beim Befehlssatz gibt es keine Unterschiede.
Alles, was Sie mit dem ATARI 400 machen können, funktioniert auch beim ATARI 800. Das umgekehrte gilt im allgemeinen auch. Hier kann es jedoch Ausnahmen geben. Der ATARI 800 bietet Ihnen einige zusätzliche Funktionen, die ihn etwas universeller als den ATARI 400 machen. Sie können zum Beispiel den Speicher des ATARI 800 ohne weiteres selbst ausbauen. Das ist beim ATARI 400 nicht möglich. Desweiteren können Sie an den ATARI 800 sowohl ein normales Fernsehgerät als auch, wenn Sie besonderen Wert auf die Bildqualität legen, einen Videomonitor anschließen. Beim ATARI 400 ist hingegen nur ein Anschluß für ein Fernsehgerät vorhanden. Außerdem hat der ATARI 800 eine schreibmaschinenähnliche Tastatur und der ATARI 400 eine Folientastatur. Und schließlich können in den ATARI 800 zwei Programmkassetten eingesteckt werden, während im ATARI 400 nur Platz für eine vorhanden ist.

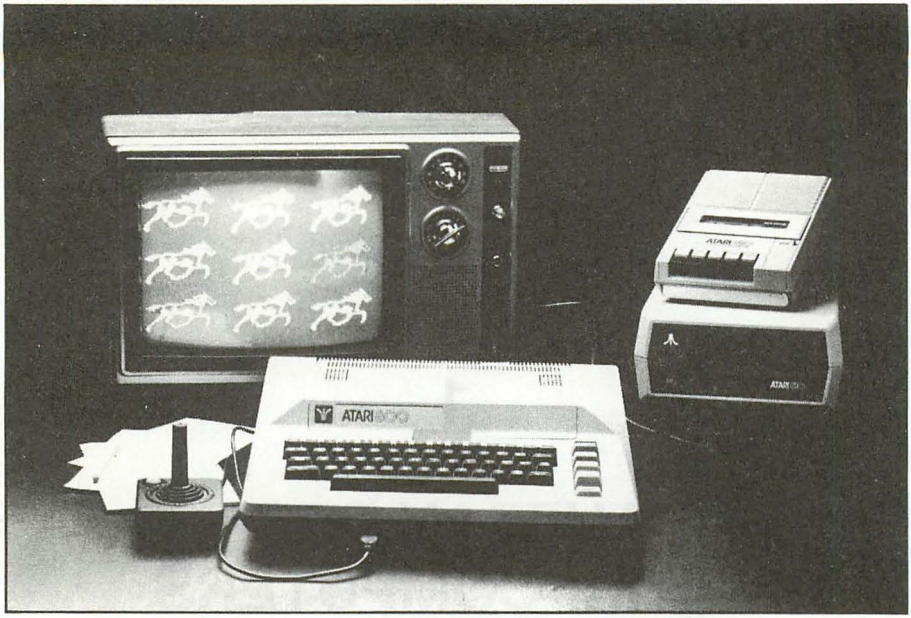


Bild 1-1: So könnte ein komplettes ATARI-Computersystem aussehen

Bild 1-2: Der ATARI 400

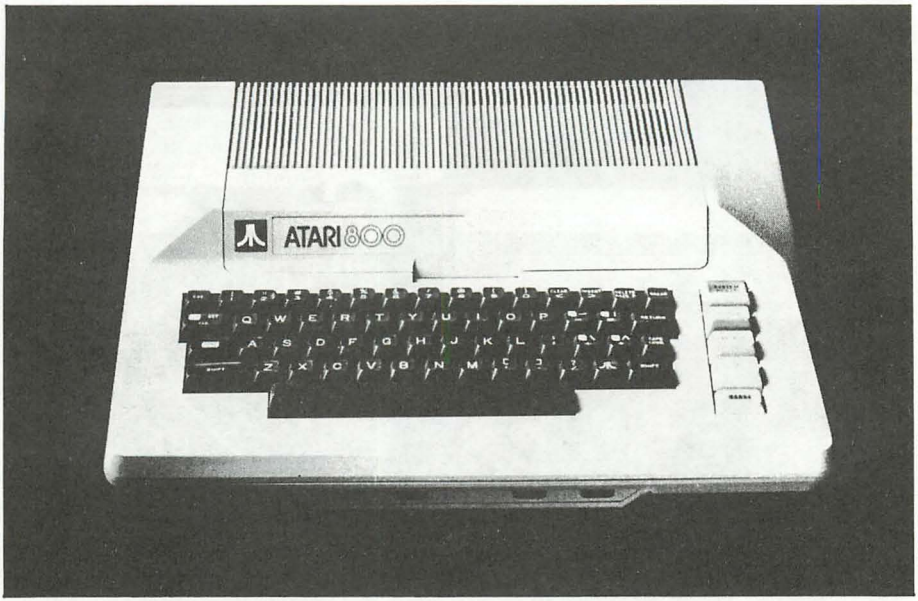


Bild 1-3: Der ATARI 800

Natürlich hat auch der ATARI 400 seine Daseinsberechtigung. Durch die versiegelte Tastatur ist sein Inneres vor Staub, Schmutz und verschütteten Flüssigkeiten geschützt. Außerdem ist er kompakter, leichter und billiger als der ATARI 800.
Wenn im folgenden vom "ATARI-Computer" gesprochen wird, dann ist damit sowohl der ATARI 400 als auch der ATARI 800 gemeint. Falls in einer Illustration etwas an einem Modell demonstriert wird, dann gilt dasselbe auch für das andere Modell. Sollte das einmal nicht der Fall sein, so ist das im begleitenden Text erwähnt.

Die Tastatur und das Fernsehgerät

Erst mit Hilfe der Tastatur und des Fernsehgerätes können Sie überhaupt mit Ihrem ATARI-Computer arbeiten. Über die Tastatur geben Sie dem Computer bestimmte Anweisungen oder Befehle. Um die Eingabe zu erleichtern, sind die Tasten der Tastatur wie auf einer Schreibmaschine angeordnet. Der ATARI 400 ist allerdings auf Grund des Aufbaus der Tastatur und der Tatsache, daß die Tastatur ziemlich kompakt ist, nicht dafür geeignet, besonders schnell zu schreiben. Bei beiden Tastaturen gibt es einige Tasten, die Sie bei einer Schreibmaschine nicht finden werden. Die Funktion dieser Tasten wird in Kapitel 2 beschrieben.
Als Sichtgerät wird normalerweise ein Farbfernsehgerät verwendet werden. An den ATARI 800 kann allerdings auch ein Farbmonitor angeschlossen werden. Auch ein Schwarz-Weiß-Fernsehgerät kann benutzt werden. An die Stelle der Farben treten dabei unterschiedliche Graustufen. Auf dem Bildschirm werden nicht nur Ihre Eingaben (die Sie so noch einmal auf ihre Richtigkeit überprüfen können), sondern auch die Antworten des Computers auf Ihre Eingaben dargestellt.
Beim Bildschirm kann zwischen mehreren Betriebsarten gewählt werden. In einer Betriebsart wird einfarbiger Text auf dem Bildschirm dargestellt. In zwei anderen Betriebsar-

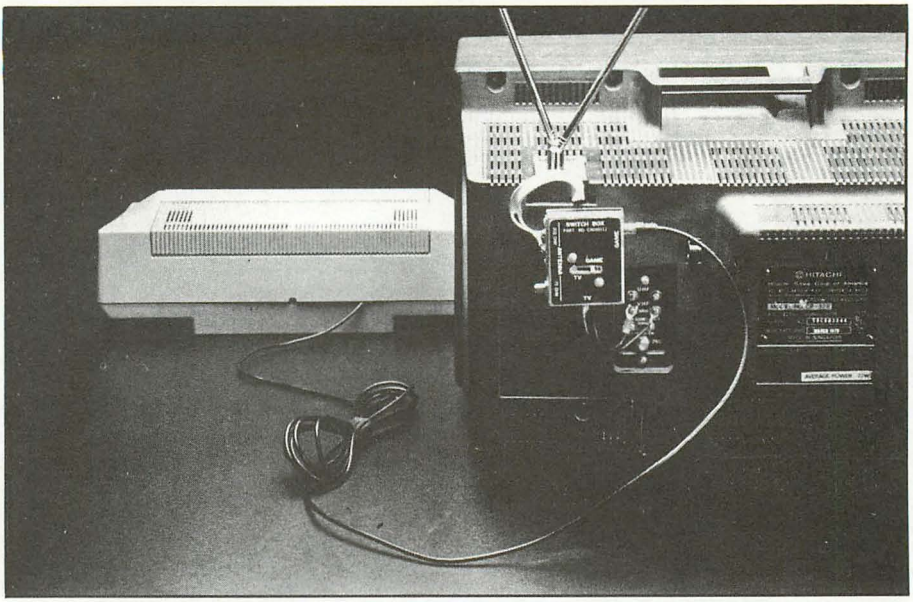


Bild 1-4: Anschluß eines Fernsehgerätes an den ATARI 400 oder 800

ten kann Text in vier verschiedenen Farben auf dem Bildschirm dargestellt werden. Andere Betriebsarten sind speziell dafür gedacht, um Grafiken zu erstellen. Bei der einfarbigen Textdarstellung wird der Bildschirm in 24 Zeilen zu je 40 Zeichen unterteilt. In den anderen Betriebsarten sieht die Unterteilung hingegen anders aus. Die Möglichkeiten der Grafikdarstellung sind in Kapitel 8 und 9 beschrieben.
Die meisten Besitzer eines ATARI-Computers werden als Sichtgerät ein Fernsehgerät verwenden, weil es in den meisten Fällen sowieso vorhanden ist. Ein etwas schärferes Bild als ein Fernsehgerät liefert allerdings ein Videomonitor.
Bei einem Fernsehgerät wird der ATARI-Computer entweder direkt oder über ein Umschaltkästchen (siehe Bild 1-4) an den Antenneneingang angeschlossen. Wenn sich der an diesem Umschaltkästchen befindliche Schalter in der einen Stellung befindet, dann ist der Computer mit dem Fernsehgerät verbunden. In der anderen Stellung ist es die Antenne. Dadurch entfällt das sonst übliche Umstecken von Kabeln.
Ein Videomonitor wird direkt an die an der rechten Seite des ATARI 800 befindlichen fünfpoligen Buchse angeschlossen.

Das Innenleben des ATARI-Computers

Unter dem Gehäuse des ATARI 400/800 verbirgt sich der eigentliche Computer, bestehend aus dem Mikroprozessor als Kernstück und verschiedenen anderen Bausteinen, mit deren Hilfe der Computer und die Peripheriegeräte überhaupt erst angesprochen und benutzt werden können. Um dieses Innenleben brauchen Sie sich im Grunde allerdings überhaupt nicht zu kümmern.
Sowohl der ATARI 400 als auch der ATARI 800 verfügen über eine Klappe, die zum Einstecken von Programmkassetten (ROM-Kassetten) geöffnet werden kann. Im ATARI 400 ist nur Platz für eine Programmkassette vorhanden, während im ATARI 800 Platz für zwei

ist (siehe Bild 1-6). Um auf die Speichermodule zugreifen zu können, ist beim ATARI 800 ein Teil der Gehäuseabdeckung abnehmbar (siehe Bild 1-7).

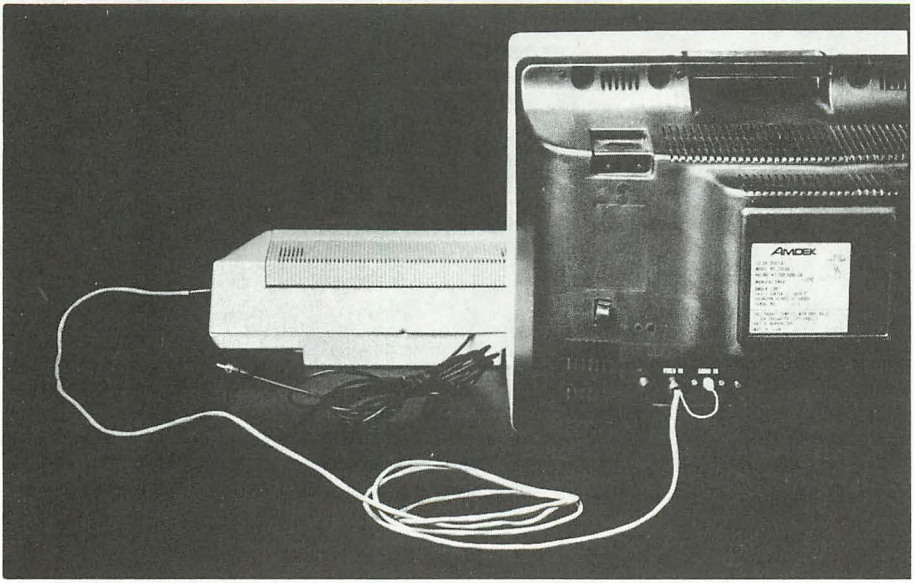


Bild 1-5: Anschluß eines Videomonitors an den ATARI 800

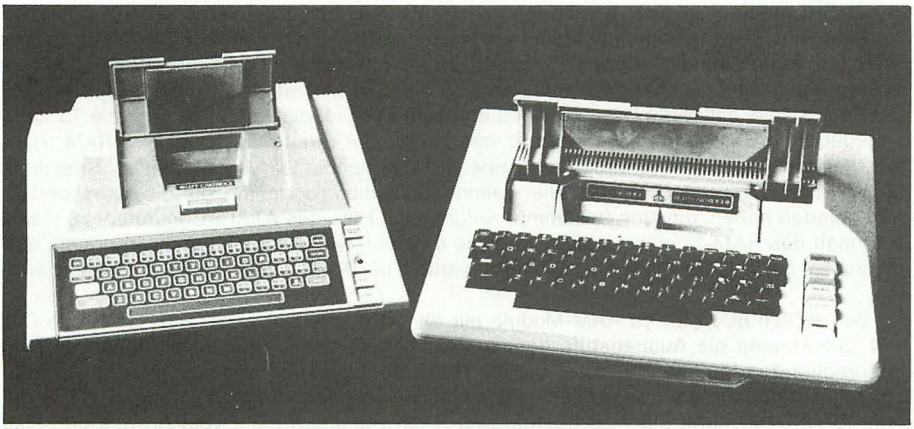


Bild 1-6: Die Steckplätze für die ROM-Kassetten

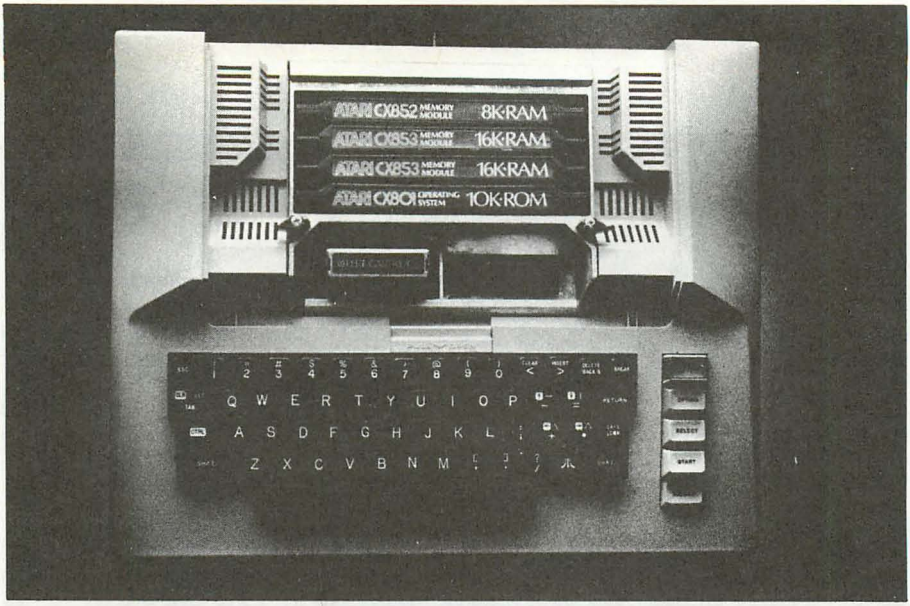


Bild 1-7: ATARI 800 mit geöffnetem Gehäuse und den Speichermodulen

Der Speicher

Die Speicherkapazität eines Computers wird in Bytes gemessen. In einem Byte kann ein Zeichen gespeichert werden. Ihr Computer verfügt, je nachdem, wie weit er ausgebaut ist, über 18432 bis 61440 Bytes. Dies ist gleichbedeutend mit 18K bis 60K, wobei 1K 1024 Bytes entspricht. Die Leistungsfähigkeit eines Computers wird, wie Sie später noch sehen werden, unter anderem auch durch die Speicherausbaustufe bestimmt.
Im ATARI-Computer gibt es, wie in den meisten Computern, zwei verschiedene Speicher. Der eine wird ROM (Read-Only Memory = Nur-Lese-Speicher) genannt. Der Inhalt dieses Speichers kann niemals verändert werden, auch nicht durch Abschalten der Spannungsversorgung. In diesem Speicher befinden sich die zum Betrieb des Computers notwendigen Programme, mit deren Hilfe der Computer zum Beispiel die von Ihnen über die Tastatur eingegebenen Befehle interpretiert und verarbeitet. Der zweite Speicher wird RAM (Random-Access Memory = Schreib/Lese-Speicher) genannt. Der Inhalt dieses Speichers kann durchaus verändert werden. Hier befinden sich die Programme, die Sie selbst erstellt oder geladen haben, um eine bestimmte Aufgabe mit Hilfe des ATARI-Computers zu lösen. Der Inhalt des RAM-Speichers bleibt solange erhalten, wie die Spannungsversorgung eingeschaltet ist. Sobald Sie sie abschalten, verliert der RAM-Speicher seinen Speicherinhalt.
Für den ATARI 800 gibt es RAM-Module mit 8K oder 16K Speicherkapazität (siehe Bild 1-8). Sie können die Ausbaustufe des RAM-Speichers selbst bestimmen, indem Sie die entsprechenden Module, nachdem Sie die Gehäuseabdeckung abgenommen haben, einfach in die dafür vorgesehenen Steckplätze einstecken (siehe Bild 1-7).
Beim ATARI 400 kann die Ausbaustufe des RAM-Speichers nicht vom Anwender selbst verändert werden. Es gibt jedoch einige Händler, die diese Änderung vornehmen können.

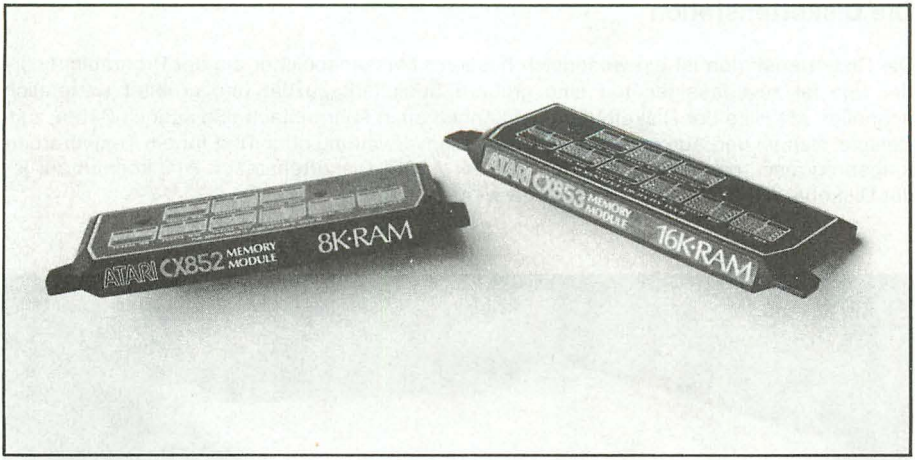


Bild 1-8: Die RAM-Module des ATARI 800

Der Programmrecorder

Der ATARI-Programmrecorder 410 (siehe Bild 1-9) wurde speziell für die ATARI-Computer gebaut. Es handelt sich hierbei keinesfalls um einen gewöhnlichen Kassettenrecorder. Mit Hilfe dieses Programmrecorders können im RAM-Speicher befindliche Programme auf Kassetten gespeichert und später wieder geladen werden. Damit können Sie ganze Programmbibliotheken aufbauen. Auf einer einzigen Kassette mit 30 Minuten Spieldauer können 51200 Zeichen gespeichert werden.

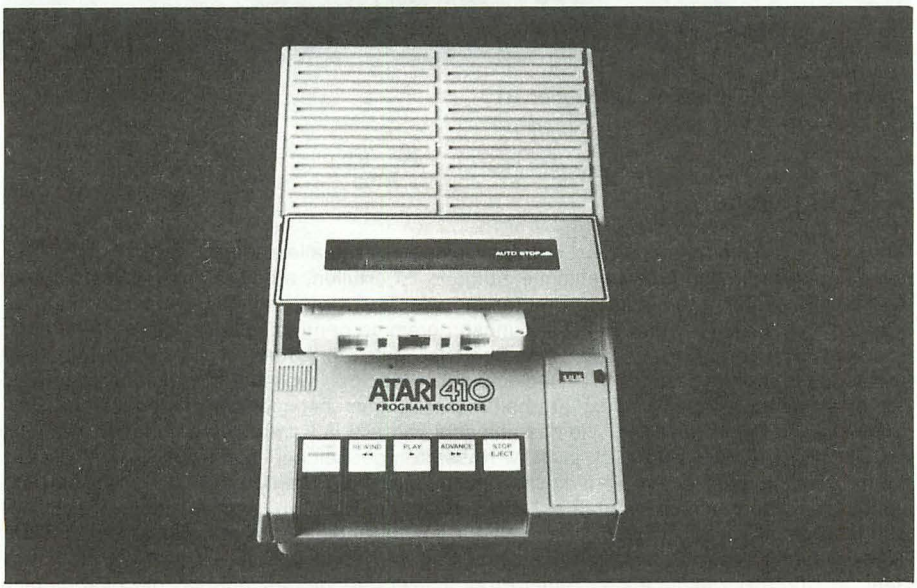


Bild 1-9: Der ATARI-Programmrecorder 410

Die Diskettenstation

Die Diskettenstation ist ein wesentlich besserer Massenspeicher als der Programmrecorder. Sie ist zuverlässiger, hat eine größere Speicherkapazität und arbeitet wesentlich schneller. Mit Hilfe der Diskettenstation können auch sehr einfach und schnell Daten, zum Beispiel Namen und Adressen für eine Adressenverwaltung oder Text für ein Textverarbeitungsprogramm, gespeichert werden. Mit der ATARI-Diskettenstation 810 können auf jeder Diskette 92160 Zeichen gespeichert werden.

Bild 1-10: Die ATARI-Diskettenstation 810

Die Programme

Die Programme sind genauso ein Teil des Computers wie alle anderen Komponenten, aus denen er besteht. Um eine bestimmte Aufgabe zu erfüllen, arbeiten im ATARI-Computer mehrere Programme zusammen. Programme wie Textverarbeitungs-, Datenverwaltungs- oder Spielprogramme werden Anwendungsprogramme genannt. Sie werden oft von einer Kassette oder Diskette in den RAM-Speicher geladen. Wenn Sie zum Beispiel Ihre Korrespondenz mit Hilfe des ATARI-Computers erledigen wollen, dann nehmen Sie eine Diskette, auf der sich ein Textverarbeitungsprogramm befindet und laden dieses Programm in den RAM-Speicher. Wie das gemacht wird, ist in Kapitel 2 beschrieben. Manche Anwenderprogramme sind auch in ROM-Kassetten gespeichert (siehe Bild 1-11), die Sie nur in den Computer einstecken müssen (siehe Bild 1-6). Bei den in ROM-Kassetten gespeicherten Programmen handelt es sich sehr oft um Spielprogramme. Wenn Sie ein bestimmtes Spiel spielen wollen, brauchen Sie nichts anderes tun, als die entsprechende ROM-Kassette in den Computer einzustecken.
Sehr oft sind Programme auch in einer Programmiersprache geschrieben, die zwar einfach für den Programmierer ist, aber vom ATARI-Computer nicht ohne weiteres verarbeitet wer-

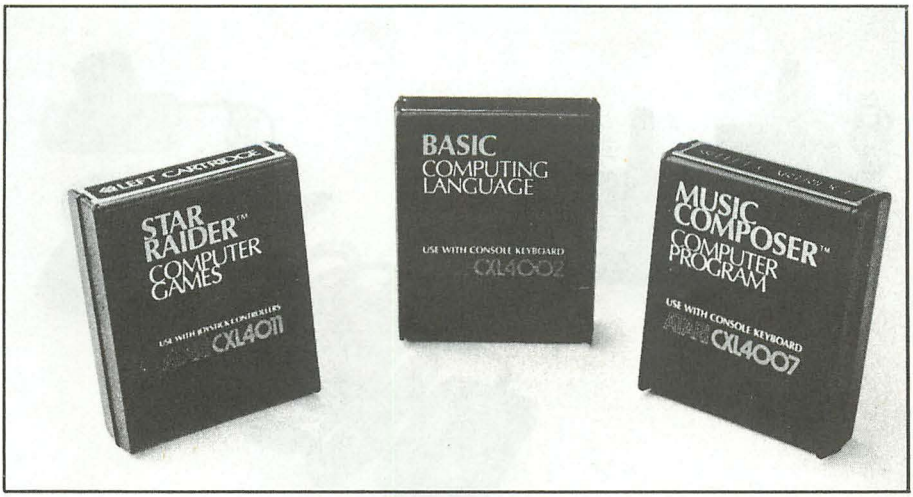


Bild 1-11: Einige ROM-Kassetten

den kann. Für die Übersetzung eines in einer bestimmten Programmiersprache geschriebenen Anwenderprogrammes in eine Sprache, die der Computer verstehen kann, sorgt ein weiteres Programm, der sogenannte Interpreter. Der Interpreter für das ATARI-BASIC ist in einer ROM-Kassette gespeichert, die einfach in den Computer eingesteckt wird.
Der Interpreter arbeitet wiederum mit einem anderen Programm, dem Operating System, zusammen. Dieses Programm steuert und koordiniert die Funktionen der verschiedenen Bestandteile des Computers. Dazu gehört unter anderem die Übertragung eines Programmes von Kassette oder Diskette in den RAM-Speicher des Computers oder das Lesen von Zeichen von der Tastatur und die Darstellung dieser Zeichen auf dem Bildschirm. Das Operating System ist im ROM gespeichert. Beim ATARI 800 befindet sich dieses ROM in einem Einsteckmodul direkt neben den Steckplätzen für die RAM-Module (siehe Bild 1-7).

Die Kontrolleranschlüsse

An der Vorderseite des ATARI-Computers befinden sich vier Anschlüsse für Steuerknüppel, Drehregler oder spezielle Kleintastaturen (siehe Bild 1-12). Diese verschiedenartigen Kontroller wurden früher meistens bei Spielen verwendet, aber heute werden sie immer häufiger auch im Zusammenhang mit anderen Programmen eingesetzt.

Die Drucker

Bei vielen Anwendungen, besonders im Geschäftsleben, müssen bestimmte Dinge zu Papier gebracht werden. Dafür stehen Ihnen drei verschiedenen ATARI-Drucker zur Verfügung. Die ATARI-Drucker 820 und 822 (siehe Bild 1-13) werden direkt an die Datenleitung des ATARI-Computers, mit der auch die Diskettenstation, der Programmrecorder usw. verbunden sind, angeschlossen. Der ATARI-Drucker 825 wird über das ATARI-Interfacemodul 850 mit dem Computer verbunden (siehe Bild 1-14). Auch andere Drucker können an dieses Interfacemodul angeschlossen werden. Es gibt Drucker für alle möglichen Schriftarten, in jeder Größe und Preislage. Das Schriftbild einiger Drucker kann sich durchaus mit dem einer Schreibmaschine messen. Andere Drucker sind wiederum besonders zum Erstellen

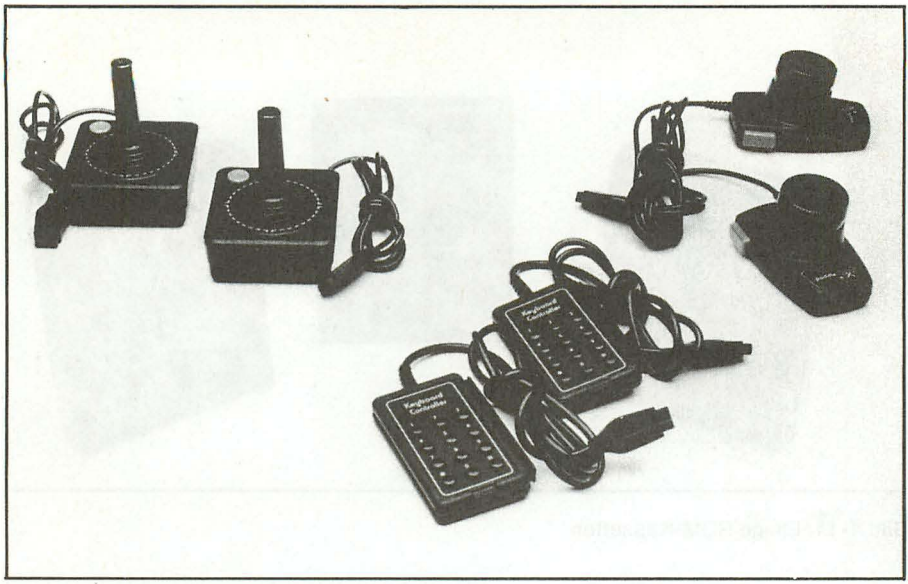


Bild 1-12: Die verschiedenen Kontroller

von Grafiken geeignet (manchmal sogar in Farbe). Und es gibt sogar Drucker, die beides können.

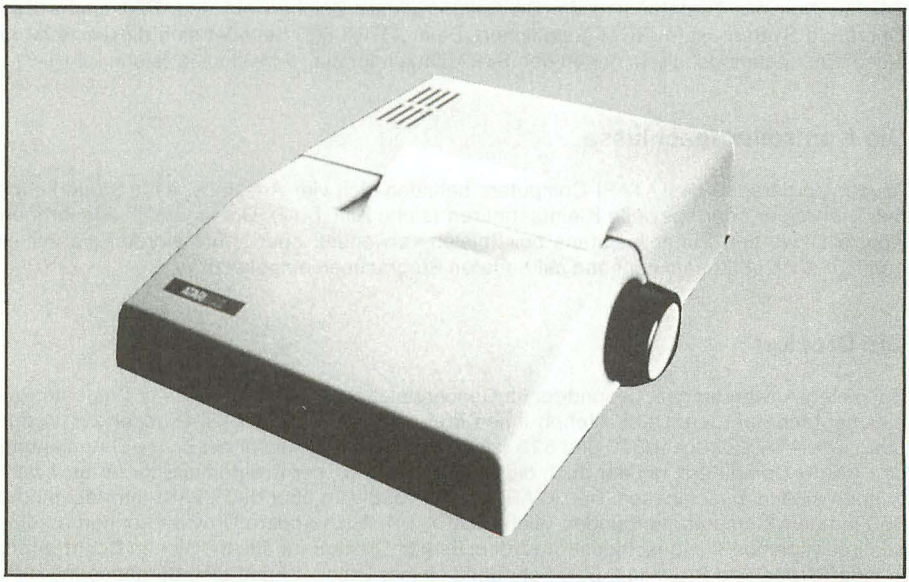


Photo courtesy of Atari, Inc.

Bild 1-13: Der ATARI-Drucker 822 (Thermodrucker)

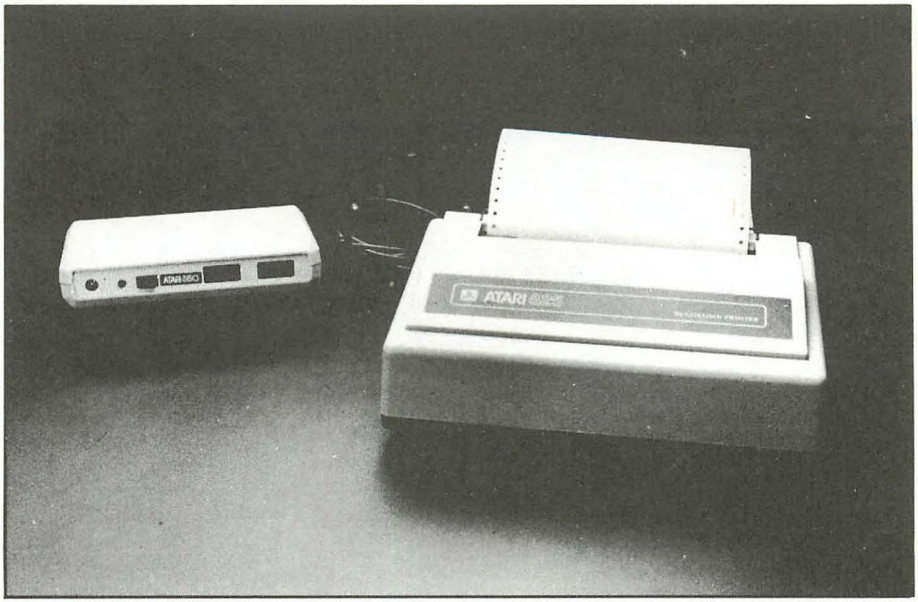
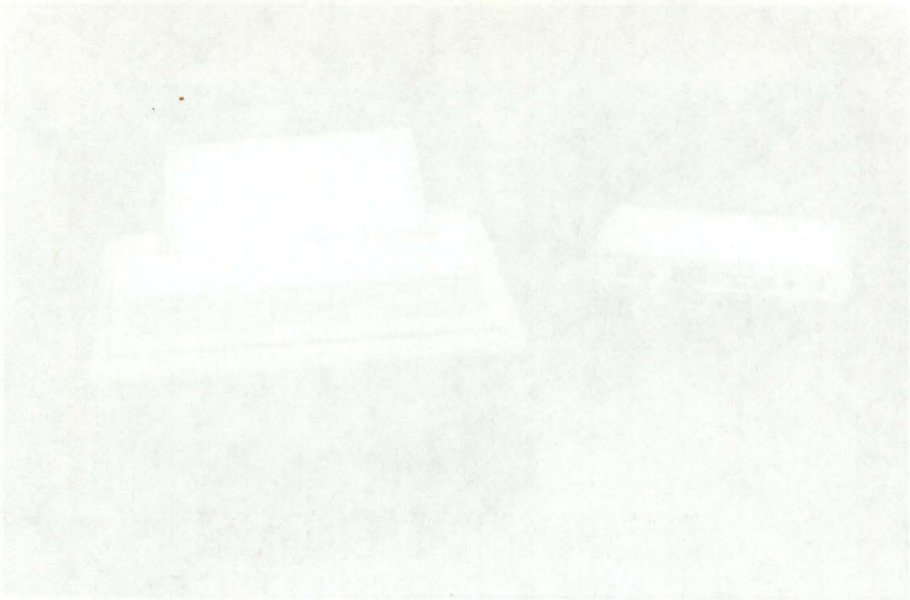


Bild 1-14: Der ATARI-Drucker 825 und das ATARI-Interfacemodul 850



2

BEDIENUNG DES ATARI-COMPUTERS

Dieses Kapitel soll Ihnen über die Schwierigkeiten hinweghelfen, die Sie vielleicht haben, wenn Sie Ihr System zum ersten Mal in Betrieb nehmen. Bevor Sie weiterlesen , vergewissern Sie sich, ob alle Bestandteile Ihres Systems ordnungsgemäß untereinander und mit der Spannungsversorgung verbunden sind. Diese Dinge sind in den Bedienungsanleitungen der einzelnen Komponenten ausführlich beschrieben. Sollten Sie trotzdem noch irgendwelche Fragen haben, setzen Sie sich mit Ihrem Händler oder einem anderen ATARI-Besitzer in Verbindung.

Einsetzen der ROM-Kassetten

Die eingesetzte ROM-Kassette bestimmt, welche Funktion der ATARI-Computer erfüllt. Nach dem Öffnen einer Klappe in der Gehäuseabdeckung kann eine ROM-Kassette eingesetzt beziehungsweise ausgewechselt werden (siehe Bild 2-1).Im ATARI 400 ist Platz für eine und im ATARI 800 Platz für zwei ROM-Kassetten vorhanden. Beim ATARI 800 wird fast immer der linke Steckplatz benutzt. Wenn Sie eine ROM-Kassette austauschen wollen, ziehen sie zunächst die in den Computer eingesteckte ROM-Kassette senkrecht nach oben heraus. Dann nehmen Sie die neue ROM-Kassette und setzen Sie wieder in den Computer ein. Die Beschriftung der Kassette muß dabei zu Ihnen zeigen. Vergewissern Sie sich noch einmal, ob die Kassette richtig eingesetzt ist und schließen Sie dann wieder die Abdeckklappe. Der Computer wird übrigens beim Öffnen der Klappe automatisch ausgeschaltet. Wenn keine ROM-Kassette eingesetzt ist, befindet sich der ATARI-Computer in der Memo-Pad-Betriebsart. Der Computer hat in diesem Fall dieselbe Funktion wie ein Notizblock: Es wird lediglich das auf dem Bildschirm dargestellt, was Sie über die Tastatur eingeben.
Für die in diesem Buch enthaltenen Funktionsbeschreibungen wird davon ausgegangen, daß die mit "BASIC Computing Language" bezeichnete ROM-Kassette eingesteckt ist.

Bild 2-1: Einsetzen einer ROM-Kassette in den ATARI 800 (beim ATARI 400 wird entsprechend vorgegangen)

Einschalten des Systemes

Bevor Sie irgendwelche Spannungsversorgungen einschalten, vergewissern Sie sich, ob alle Bestandteile des Systemes ordnungsgemäß miteinander verbunden sind. In Bild 2-2 ist ein voll ausgebautes Computersystem mit allen notwendigen Verbindungen schematisch dargestellt.
Beim Einschalten der einzelnen Komponenten eines solchen Systemes muß eine bestimmte Reihenfolge eingehalten werden:

1. Schalten Sie das Fernsehgerät ein. Falls Sie ein Umschaltkästchen benutzen, muß der Computer auf den Antenneneingang geschaltet sein.
2. Falls Sie sie benötigen, schalten Sie nun die Diskettenstation ein und legen Sie eine Diskette ein, auf der das Disk Operating System gespeichert ist.
3. Falls Sie ein Gerät benutzen wollen, das an das ATARI-Interfacemodul 850 angeschlossen wird, schalten Sie nun das Interfacemodul ein.
4. Als nächstes schalten Sie den ATARI 400/800 ein und stimmen Sie das Fernsehgerät so ab, daß das Bild- und Tonsignal des Computers einwandfrei empfangen wird.
5. Falls Sie einen Drucker benutzen wollen, schalten Sie ihn jetzt ein. Beim ATARI-Drucker 825 muß außerdem auch das ATARI-Interfacemodul 850 eingeschaltet sein (siehe oben).

Wenn Sie diese Reihenfolge nicht einhalten, kann es unter Umständen vorkommen, daß der ATARI-Computer auf eine Komponente des Systemes nicht zugreifen kann. Die oben aufgeführten Schritte werden im folgenden ausführlich beschrieben.

Schritt 1: Das Fernsehgerät

Als erstes schalten sie das Fernsehgerät beziehungsweise den Monitor ein. Solange das Fernsehgerät noch nicht eingestellt oder der Computer selbst noch nicht einge-

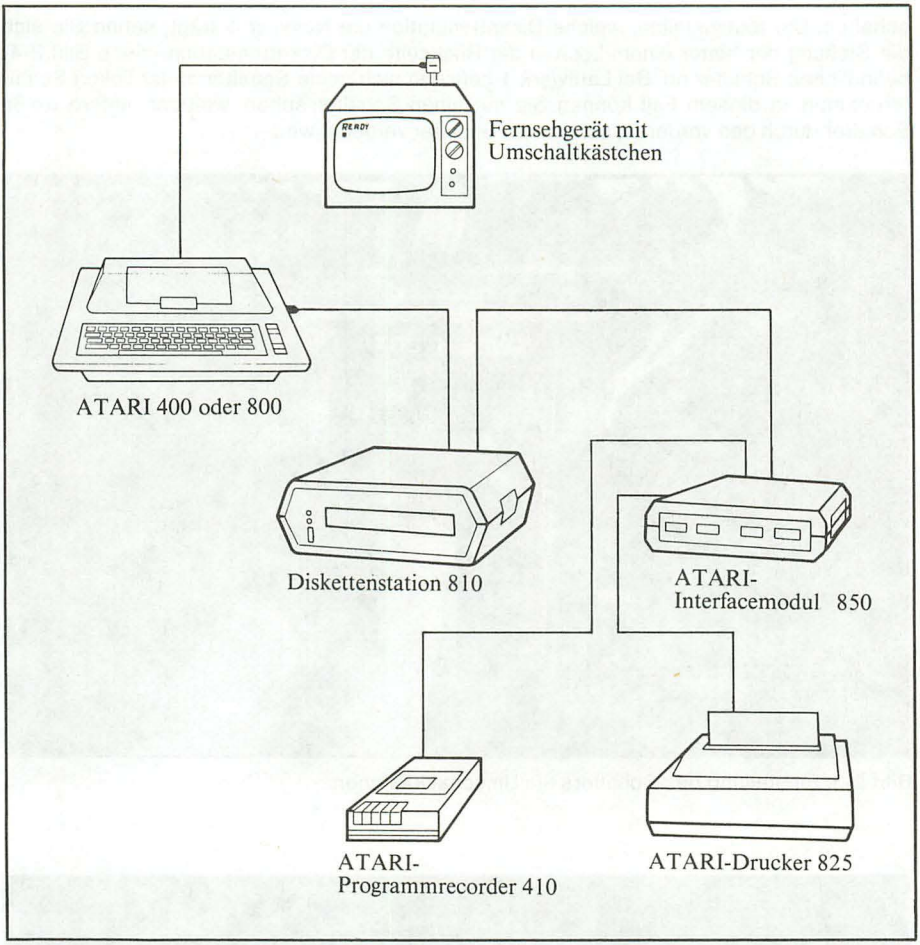


Bild 2-2: Verbindung der einzelnen Komponenten eines voll ausgebauten Systems.

schaltet ist, kann kein Signal empfangen werden und das Gerät rauscht. Deshalb drehen Sie die Lautstärke am Amfang am besten etwas herunter. Falls Sie ein Umschaltkästchen benutzen (nur beim Fernsehgerät), muß der Computer auf den Antenneneingang geschaltet sein. Die entsprechende Schalterstellung ist meistens mit "Computer" oder "Game" gekennzeichnet (siehe Bild 2-3).

Schritt 2: Die Diskettenstation

Wenn sie keine Diskettenstation haben oder sie (zunächst) nicht benutzen wollen, dann können Sie diesen Abschnitt überschlagen. Anderenfalls schalten Sie die Diskettenstation jetzt ein. Daraufhin wird der Motor der Diskettenstation anlaufen und die an der Vorderseite der Diskettenstation befindlichen Lämpchen aufleuchten. Nach einigen Sekunden wird der Motor der Diskettenstation wieder stehenbleiben und nur noch das mit "PWR ON" bezeichnete Lämpchen leuchten.
Wenn Sie mehrere Diskettenstationen haben, müssen Sie an dieser Stelle Laufwerk 1 ein-

schalten. Um festzustellen, welche Diskettenstation die Nummer 1 trägt, sehen Sie sich die Stellung der hinter einem Loch in der Rückseite der Diskettenstation (siehe Bild 2-4) befindlichen Schalter an. Bei Laufwerk 1 befinden sich beide Schalter in der linken Schalterposition. In diesem Fall können Sie nur einen Schalter sehen, weil der hintere weiße Schalter durch den vorderen, schwarzen Schalter verdeckt wird.

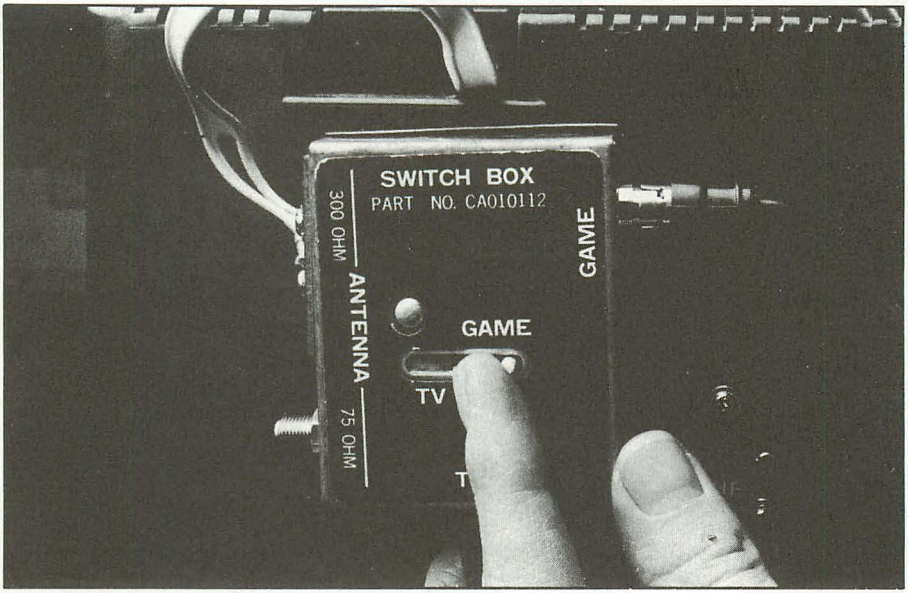


Bild 2-3: Einstellung des Schalters am Umschaltkästchen

Bild 2-4: Die Schalter zur Festlegung der Laufwerksnummer

Legen Sie nun eine mit "Disk File Manager Master Copy" oder "Disk File Manager II Master Copy" bezeichnete Diskette beziehungsweise eine Kopie davon in Laufwerk 1 ein. Die Diskette muß ganz in die Diskettenstation eingeschoben werden (mit dem Etikett nach oben), bevor die Tür des Laufwerkes geschlossen wird. Außer den oben angegebenen Disketten können Sie an dieser Stelle auch eine andere Diskette einlegen, die laut der dazugehörigen Beschreibung beim Einschalten des Computers eingelegt werden soll. Mehr über den Umgang mit Disketten erfahren Sie in dem in diesem Kapitel enthaltenen Abschnitt über die Bedienung der Diskettenstation.

Schritt 3: Das ATARI-Interfacemodul 850

Schalten Sie das ATARI-Interfacemodul 850 (siehe Bild 2-5) nur dann ein, wenn Sie ein Gerät benutzen wollen, das an das Interfacemodul angeschlossen ist. Anderenfalls können Sie direkt zu Schritt 4 übergehen.

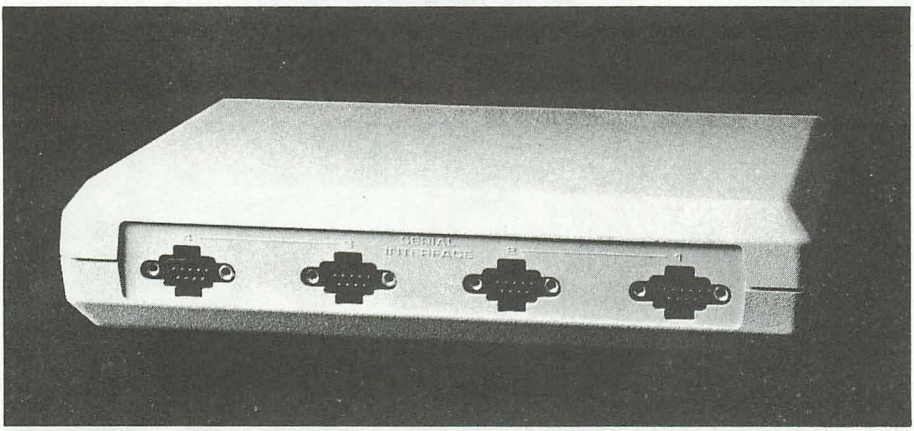


Bild 2-5: Die seriellen Schnittstellen des ATARI-Interfacemoduls 850

Schritt 4: Der ATARI 400/800

Öffnen Sie zunächst die Klappe der Gehäuseabdeckung und überzeugen Sie sich davon, daß die richtige ROM-Kassette eingesteckt ist (siehe Bild 2-1). Danach schließen Sie die Abdeckklappe wieder und überprüfen ein letztes Mal die Verbindungen zwischen den einzelnen Komponenten Ihres Computersystemes (siehe Bild 2-2). Schalten Sie nun den Computer mit Hilfe des direkt neben dem Anschluß für die Spannungsversorgung befindlichen Schalters (siehe Bild 2-6) ein (Schalterstellung "ON") und drehen Sie den Lautstärkeregler Ihres Fernsehgerätes ein wenig auf. Die Netzkontrollampe des Computers muß nun leuchten. Falls Sie den Computer nicht zum ersten Mal einschalten und das Fernsehgerät bereits eingestellt ist (siehe unten) oder wenn Sie einen Monitor verwenden, werden Sie auf dem Bildschirm ein blaues Feld mit einer schwarzen Umrandung sehen und eine Art Brummen hören (wenn der Lautstärkeregler weit genug aufgedreht ist). Falls die Diskettenstation eingeschaltet ist, wird der Motor der Diskettenstation anlaufen. Kurz darauf wird in weißen Buchstaben "READY" auf dem Bildschirm erscheinen (siehe Bild 2-7) und der Motor der Diskettenstation stehenbleiben.

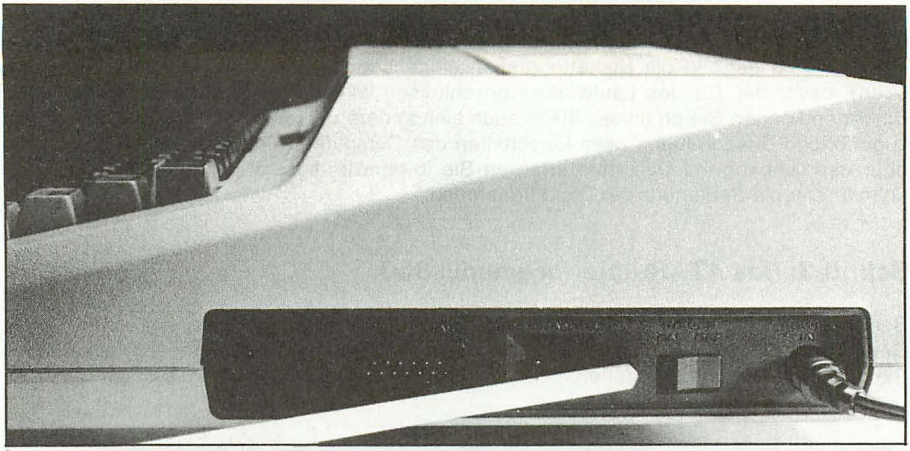


Bild 2-6: Der Netzschalter des ATARI 800 (beim ATARI 400 ähnlich)

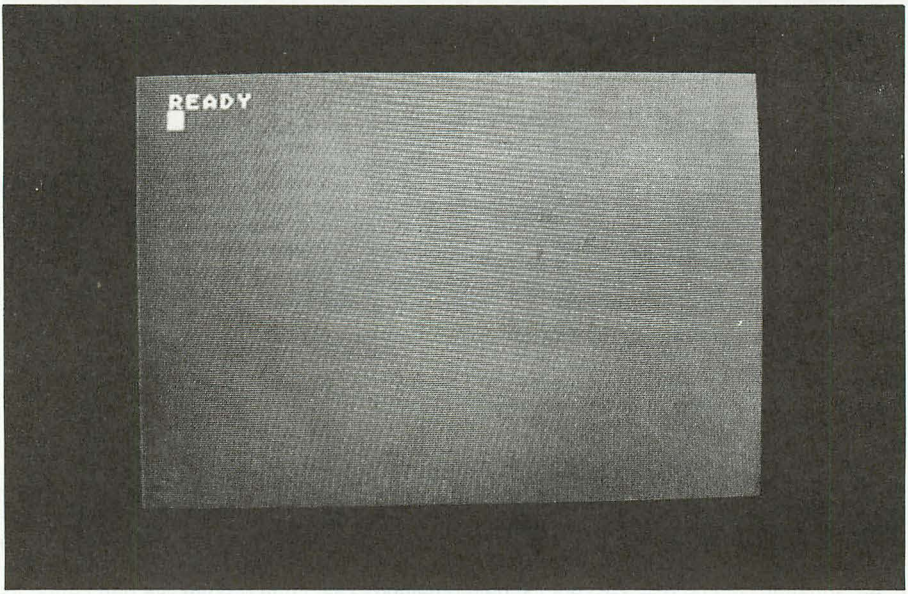


Bild 2-7: Diese Meldung erscheint nach dem Einschalten des Computers auf dem Bildschirm

Wenn Sie ein Fernsehgerät verwenden und den Computer zum ersten Mal eingeschaltet haben, müssen Sie nun das Fernsehgerät einstellen. Suchen Sie dazu im VHF-Bereich das Bild- und Tonsignal des Computers. Mit Hilfe eines an der rechten Seite des Computers befindlichen Schiebeschalters (siehe Bild 2-8) können Sie dabei zwischen Kanal 2 und Kanal 3 wählen. Möglicherweise ist der Bild- und Tonempfang in einem Kanal besser. Nachdem Sie das Signal gefunden haben, belegen Sie am besten eine Programmtaste Ihres Fernsehgerätes damit.

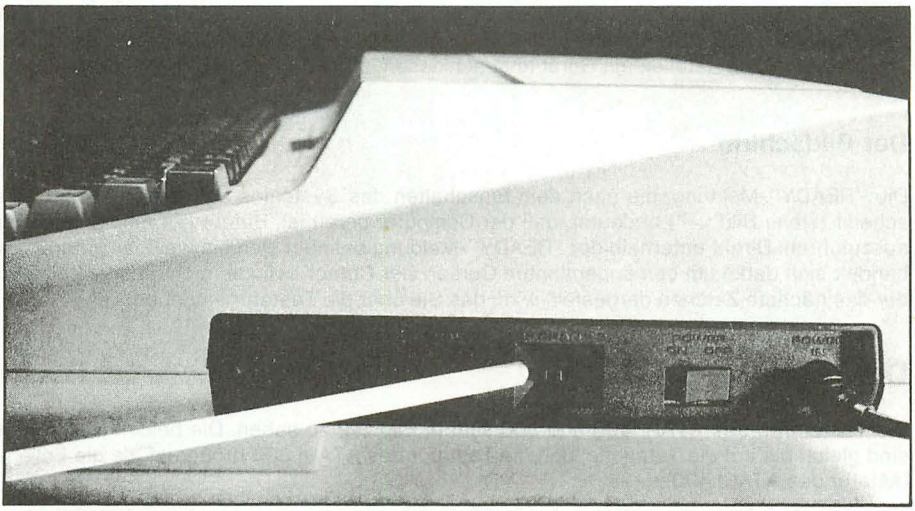


Bild 2-8: Der Schalter zur Wahl des Kanals für das Bild- und Tonsignal des ATARI-Computer

Falls 30 Sekunden nach dem Einschalten des Computers noch nicht "READY" auf dem Bildschirm erschienen ist, liegt irgendein Fehler vor. Schalten Sie alle Geräte aus, überprüfen Sie alle Verbindungsleitungen und versuchen Sie dann das Ganze noch einmal. Wenn Sie eine Diskettenstation benutzen, vergewissern Sie sich, daß die richtige Diskette mit dem Etikett nach oben eingelegt ist und daß die Tür der Diskettenstation geschlossen ist. Wenn das nicht der Fall ist, wird die Diskettenstation ab und zu ein kurzes Schnarren von sich geben und/oder die Meldung "BOOT ERROR" auf dem Bildschirm erscheinen.

Wenn nach dem erneuten Einschalten noch immer nicht "READY" auf dem Bildschirm erscheint, schalten Sie wieder alle Geräte aus und holen sich von jemandem Hilfe, der mehr Erfahrung hat (zum Beispiel von Ihrem Händler).

Schritt 5: Der Drucker

Nun, nachdem Sie alle anderen Schritte erledigt haben, können Sie den Drucker einschalten, wenn Sie ihn brauchen und ausschalten, wenn Sie ihn nicht mehr brauchen. Beim ATARI-Drucker 825 muß außerdem das ATARI-Interfacemodul 850 eingeschaltet sein.

Ein- und Ausschalten von einzelnen Komponenten Ihres Computersystemes

Solange Sie mit Ihrem System arbeiten wollen, muß der ATARI 400/800 selbst natürlich immer eingeschaltet sein. Das Fernsehgerät, die Diskettenstation oder den Drucker können Sie, wenn Sie Ihr System einmal gemäß der obigen Beschreibung eingeschaltet haben, ohne weiteres aus- und wieder einschalten. Wenn Sie eines dieser Geräte benutzen wollen, muß es natürlich eingeschaltet sein. Auch das ATARI-Interfacemodul 850 können

Sie nach Belieben aus- und einschalten, sofern der ATARI-Drucker 825 als einziges daran angeschlossen ist. Das Interfacemodul darf jedoch nicht ausgeschaltet werden, wenn an die seriellen Schnittstellen ein Gerät angeschlossen ist.

Der Bildschirm

Die "READY"-Meldung, die nach dem Einschalten des Systemes auf dem Bildschirm erscheint (siehe Bild 2-7) bedeutet, daß der Computer bereit ist, Befehle zu empfangen und auszuführen. Direkt unterhalb der "READY"-Meldung befindet sich ein weißes Quadrat. Es handelt sich dabei um den sogenannten Cursor. Der Cursor befindet sich an der Stelle, an der das nächste Zeichen dargestellt wird, das Sie über die Tastatur eingeben.

Die Tastatur

Die Tastaturen des ATARI 400 und 800 sind in Bild 2-9 zu sehen. Die beiden Tastaturen sind gleich bis auf die Tatsache, daß die Tastatur des ATARI 800 größer ist als die Folientastatur des ATARI 400.

Bild 2-9: Die Tastaturen des ATARI 400 und 800

Die Tastaturen sehen im wesentlichen genau so aus wie die Tastatur einer Schreibmaschine. Zusätzlich gibt es hier allerdings einige Tasten, die Sie bei einer Schreibmaschinentastatur nicht finden werden. Zwei davon befinden sich an der linken Seite der Tastatur und sind mit ESC und CTRL bezeichnet. Die anderen befinden sich an der rechten Seite und sind mit BREAK, CAPS/LOWR und ▲ bezeichnet. Einige der anderen Tasten sind zusätzlich mit Wörtern und Symbolen bezeichnet, die Sie ebenfalls bei einer Schreibmaschinentastatur nicht finden werden. Und schließlich befindet sich rechts von der Tastatur noch eine Reihe mit vier gelben Tasten für Sonderfunktionen.

Nehmen Sie sich ein paar Minuten Zeit und experimentieren Sie ein wenig mit der Tastatur. Probieren Sie einfach einmal aus, welche Wirkung es hat, wenn Sie eine bestimmte Taste oder auch zwei Tasten gleichzeitig drücken. Falsch machen können Sie dabei nichts. Schlimmstenfalls müßten Sie den Computer aus- und wieder einschalten.

Zeichenwiederholung

Drücken Sie eine Taste, zum Beispiel die mit G bezeichnete, für eine längere Zeit. Zunächst wird ein G und dann, nach etwa einer Sekunde, weitere G's auf dem Bildschirm dargestellt. Diese automatische Zeichenwiederholung funktioniert bei jeder Taste der Tastatur bis auf die SHIFT- und die BREAK-Taste sowie die vier gelben Sonderfunktionstasten.

Die Zeilenlänge

Normalerweise ist der Bildschirm in 24 Zeilen zu je 40 Zeichen unterteilt. Die beiden ersten Spalten einer Zeile liegen jedoch außerhalb der linken Bildschirmbegrenzung, so daß in jede Zeile nur 38 Zeichen geschrieben werden können.

Die SYSTEM-RESET-Taste

Die SYSTEM-RESET-Taste ist eine der vier gelben Sonderfunktionstasten, die sich rechts von der eigentlichen Tastatur befinden. Wenn Sie diese Taste drücken, unterbricht der Computer die Aufgabe, die er gerade erledigt, ganz gleich, worum es sich dabei handelt. Anschließend können Sie über die Tastatur neue Befehle eingeben.
Es kann allerdings Probleme geben, wenn sie die SYSTEM-RESET-Taste zu einem Zeitpunkt drücken, wenn gerade die Diskettenstation aktiviert ist. Deshalb sollten Sie immer darauf achten, daß Sie diese Taste nicht versehentlich drücken.

Die RETURN-Taste

Alles, was Sie über die Tastatur eingeben, wird zwar auf dem Bildschirm dargestellt und auch im Speicher des ATARI-Computers abgelegt, aber es wird nicht verarbeitet. Das geschieht erst dann, wenn Sie die RETURN-Taste drücken. Sie kennzeichnen damit den Abschluß einer Zeile. Nachdem die RETURN-Taste gedrückt wurde, untersucht der Computer die gerade eingegebene Zeile und gibt, wenn er sie nicht verarbeiten kann, eine Fehlermeldung aus.

Die BREAK-Taste

Wenn Sie die BREAK-Taste während der Eingabe eines Befehles drücken, ignoriert der Computer alles, was Sie vorher eingegeben haben.
Während der Ausführung eines Programmes sollten Sie die BREAK-Taste nicht drücken, weil dann die Ausführung dieses Programmes unterbrochen wird. Es gibt zwar einige Programme, bei denen das verhindert wird, aber weitaus die meisten Programme werden unterbrochen. Die Programmausführung wird zwar fortgesetzt, wenn Sie "CONT" eingeben und die RETURN-Taste drücken, aber das mindeste, was dabei zurückbleibt, ist eine zerstörte Bildschirmdarstellung.

Die SHIFT-Taste

Nach dem Einschalten des ATARI-Computers werden beim Drücken einer Buchstabentaste immer Großbuchstaben auf dem Bildschirm dargestellt. Die SHIFT-Taste hat zwar keinen Einfluß auf die Funktion der Buchstabentasten, aber die Funktion von einigen anderen Tasten kann damit sehr wohl beeinflußt werden. Mit diesen Tasten können zwei verschiedene Zeichen erzeugt werden: eines, wenn Sie die Taste allein und das zweite, wenn Sie die Taste selbst und zusätzlich die SHIFT-Taste drücken. Diese Zeichen sind auch auf den entsprechenden Tasten dargestellt. Das obere Zeichen ist das, zu dessen Erzeugung Sie zusätzlich die SHIFT-Taste drücken müssen. Einige solcher Tastenkombinationen sind in Tabelle 2-1 aufgeführt. Eine vollständige Zusammenstellung dieser Tastenkombinationen finden Sie in Teil D des Anhanges.
Im folgenden wird, wenn zusätzlich zu einer Taste die SHIFT-Taste gedrückt werden soll, vor die Bezeichnung dieser Taste der Zusatz SHIFT- gesetzt werden. SHIFT-3 bedeutet zum Beispiel, daß gleichzeitig die SHIFT-Taste und die 3 gedrückt werden soll. Das Ergebnis ist ein #-Zeichen.

Tabelle 2-1: Einige Tastenkombinationen mit der SHIFT-Taste

Tastenkombination	Funktion
SHIFT-TAB	Setzen eines Tabulators-Stops
SHIFT-<	Bildschirm löschen
SHIFT->	Zeile einfügen
SHIFT-BACK S	Zeile löschen

Die CTRL-Taste

CTRL ist eine Abkürzung für Control. Die CTRL-Taste wird wie die SHIFT-Taste immer im Zusammenhang mit einer anderen Taste benutzt. Sie müssen also zuerst die CTRL-Taste drücken und dann eine andere Taste betätigen. Im folgenden wird, wenn zusätzlich zu einer Taste die CTRL-Taste gedrückt werden soll, vor die Bezeichnung dieser Taste der Zusatz CTRL- gesetzt werden. CTRL-B bedeutet zum Beispiel, daß gleichzeitig die CTRL-Taste und B gedrückt werden soll.
Wie die SHIFT-Taste gibt auch die CTRL-Taste einigen Tasten eine Doppelfunktion. Eini-

Tabelle 2-2: Einige Tastenkombinationen mit der CTRL-Taste

Tastenkombination	Funktion
CTRL-TAB	Tabulator-Stop löschen
CTRL--	Cursor um eine Zeile nach oben bewegen
CTRL-=	Cursor um eine Zeile nach unten bewegen
CTRL-+	Cursor um eine Stelle nach links bewegen
CTRL-*	Cursor um eine Stelle nach rechts bewegen
CTRL-1	Bildschirmausgabe stoppen/wieder fortsetzen
CTRL-3	Führt normalerweise zu einer Fehlermeldung
CTRL-<	Bildschirm löschen
CTRL->	Einfügen einer Leerstelle
CTRL-BACK S	Löschen eines Zeichens
CTRL-CAPS/LOWR	Umschaltung der Tastatur auf Grafikzeichen

ge dieser Funktionen sind auch auf den entsprechenden Tasten in umgekehrter Zeichendarstellung (schwarz auf weiß) vermerkt. Mit CTRL-TAB wird zum Beispiel ein Tabulator-Stop gelöscht (CLR = Clear = Löschen). Wird die CTRL-Taste im Zusammenhang mit einer Buchstabentaste gedrückt, so wird ein Grafikzeichen erzeugt. Einige der Tastenkombinationen mit der CTRL-Taste sind in Tabelle 2-2 aufgeführt. Eine vollständige Zusammenstellung dieser Tastenkombination finden Sie in Teil D des Anhanges.

Die CAPS/LOWR-Taste

Nach dem Einschalten des ATARI-Computers werden beim Drücken einer Buchstabentaste immer Großbuchstaben auf dem Bildschirm dargestellt, ganz gleich, ob gleichzeitig die SHIFT-Taste gedrückt wurde oder nicht. Mit Hilfe der CAPS/LOWR-Taste kann auf Groß/Kleinschreibung umgeschaltet werden. Wenn Sie jetzt eine Buchstabentaste drükken, werden Kleinbuchstaben auf dem Bildschirm dargestellt. Um Großbuchstaben zu erzeugen, müssen Sie nun gleichzeitig mit einer Buchstabentaste die SHIFT-Taste drücken. Wenn Sie wieder den vorherigen Zustand herstellen wollen (nur Großbuchstaben), müssen Sie gleichzeitig die SHIFT- und die CAPS/LOWR-Taste drücken. Wenn an Stelle der Buchstaben Grafikzeichen erzeugt werden sollen, müssen Sie gleichzeitig die CRTL- und die CAPS/LOWR-Taste betätigen.

Die ⩚-Taste

Mit Hilfe der ⩚-Taste kann zwischen normaler und invertierter Zeichendarstellung umgeschaltet werden. Bei der invertierten Zeichendarstellung werden nicht weiße Zeichen auf blauem Grund, sondern blaue Zeichen auf weißem Grund dargestellt.

Die CURSOR-CONTROL-Tasten

Auf vier Tasten der Tastatur sind Pfeile dargestellt. Wenn Sie eine dieser Tasten gleichzeitig mit der CTRL-Taste drücken, wird der Cursor in die entsprechende Richtung bewegt.
Diese Funktion wird sich als sehr nützlich erweisen, weil Sie damit Eingabefehler sehr einfach verbessern können, auch dann, wenn Sie bereits weiter zurückliegen. Mit Hilfe der ← Taste wird der Cursor um eine Stelle nach links bewegt. Versuchen Sie es einmal: Tippen Sie irgendein Wort ein (zum Beispiel PRINT) und betätigen Sie dann einige Male die ← Taste. Sie werden sehen, wie der Cursor von rechts nach links über das gerade eingegebene Wort hinwegwandert. Beachten Sie, daß dabei keine Zeichen gelöscht werden. Bewegen Sie den Cursor auf diese Weise bis an die linke Bildschirmbegrenzung. Wenn Sie jetzt noch einmal die ← Taste drücken, springt der Cursor an die rechte Bildschirmbegrenzung derselben Zeile.
Mit Hilfe der → Taste wird der Cursor nicht nach links, sondern nach rechts bewegt. Auch dabei werden keine Zeichen gelöscht. Wenn sich der Cursor an der rechten Bildschirmbegrenzung befindet und die → Taste gedrückt wird, springt er an die linke Bildschirmbegrenzung derselben Zeile.
Eine ähnliche Funktion haben die ↑ und die ↓ Taste. Hiermit wird der Cursor um eine Zeile nach oben beziehungsweise nach unten bewegt. Wenn sich der Cursor am oberen Bildschirmrand befindet und die ↑ Taste betätigt wird, springt er an den unteren Bildschirmrand. Umgekehrt gilt dasselbe: Wenn sich der Cursor am unteren Bildschirmrand befindet und die ↓ Taste betätigt wird, springt er an den oberen Bildschirmrand.

Die BACK S-Taste

Jedesmal, wenn sie die BACK S-Taste drücken, wird das an der momentanen Cursorposi-

tion befindliche Zeichen gelöscht und der Cursor um eine Stelle nach links bewegt. An der linken Bildschirmbegrenzung stößt der Cursor an: Wenn sich der Cursor an der linken Bildschirmbegrenzung befindet und erneut die BACK S-Taste betätigt wird, hat das keine Wirkung.

Die CLEAR-Taste

Mit Hilfe von CTRL-< oder SHIFT-< kann der Bildschirm gelöscht werden. Danach befindet sich der Cursor in der oberen linken Ecke des Bildschirms.

Die INSERT- und die DELETE-Taste

Gleichzeitig mit der INSERT- oder der DELETE-Taste muß die SHIFT- oder die CTRL-Taste gedrückt werden. Mit CTRL-> wird rechts vom Cursor ein Leerzeichen eingefügt und mit CTRL-BACK S wird das rechts vom Cursor stehende Zeichen gelöscht. Die Position des Cursors ändert sich in beiden Fällen nicht.
Mit SHIFT-> wird über der Zeile, in der sich der Cursor gerade befindet, eine Leerzeile eingefügt. Dabei wird der ganze Bildschirminhalt ab der Zeile, in der sich der Cursor befindet, um eine Zeile nach unten verschoben. Mit SHIFT-BACK S wird die Zeile, in der sich der Cursor befindet, gelöscht. Alle darunter stehenden Zeilen werden um eine Zeile nach oben verschoben.

Die TAB-Taste

Wenn Sie die TAB-Taste allein drücken, springt der Cursor an die Stelle, wo sich der nächste Tabulator-Stop befindet. Normalerweise, also auch beim Einschalten des Computers, haben die Tabulator-Stops einen Abstand von acht Zeichen zueinander. Da sich die linke Bildschirmbegrenzung normalerweise in einem Abstand von zwei Zeichen zum linken Bildschirmrand befindet, hat der erste Tabulator-Stop nur einen Abstand von sechs Zeichen zu der linken Bildschirmbegrenzung. Mit SHIFT-TAB wird ein neuer Tabulator-Stop an die Stelle gesetzt, an der sich der Cursor gerade befindet und mit CTRL-TAB wird der Tabulator-Stop an der Stelle, an der sich der Cursor gerade befindet, gelöscht.

Die ESC-Taste

ESC ist eine Abkürzung für Escape. Es handelt sich hierbei um ein Relikt aus der Zeit, als für die Datenein- und -ausgabe bei Computern Fernschreiber verwendet wurden. Die ESC-Taste muß nicht, wie die SHIFT- oder die CTRL-Taste, gleichzeitig mit einer anderen Taste gedrückt werden, sondern bevor die andere Taste gedrückt wird.
Mit Hilfe der ESC-Taste kann die Funktion einer bestimmten Taste, zum Beispiel der CLEAR-Taste (SHIFT-<), in ein Programm eingebaut werden. Eine vollständige Beschreibung der ESC-Taste finden Sie in Kapitel 4.

Die anderen Tasten

Die Funktion der übrigen Tasten der Tastatur dürfte wohl kein Problem darstellen. Hiermit können Buchstaben, Zahlen und verschiedene, allgemein übliche Symbole dargestellt werden.
Bei vielen Schreibmaschinen gibt es keinen Unterschied zwischen der Zahl 0 und dem Buchstaben "O" oder zwischen der Zahl Eins und dem Buchstaben "l". Der ATARI-Com-

puter kennt diesen Unterschied jedoch sehr wohl. Wenn eine Zahl eingegeben werden muß, dürfen wirklich nur Ziffern und keine Buchstaben verwendet werden. Zur besseren Unterscheidung der Zahl 0 und dem Bustaben "O" wird die Null immer mit einem Querstrich auf dem Bildschirm dargestellt.

Die Bedienung des ATARI-Programmrecorders 410

Mit Hilfe des ATARI-Programmrecorders 410 können Sie auf Kassette gespeicherte Programme laden. Solche Programmkassetten können Sie kaufen oder auch selber anfertigen (siehe Kapitel 3).

Der Umgang mit Kassetten

Behandeln Sie Ihre Kassetten sorgfältig. Sie könnten sie sonst leicht beschädigen und dadurch einen Schaden verursachen, der nicht leicht zu beheben ist. Vermeiden Sie es, das Band selbst zu berühren. Es wird dabei immer etwas von den natürlichen Hautfetten zurückbleiben, ganz gleich, wie sauber Ihre Hände sind. Bewahren Sie Ihre Kassetten immer nur mit Schutzhülle auf. Lagern Sie sie nicht zu heiß und setzen Sie sie nicht der direkten Sonnenbestrahlung oder magnetischen Feldern,wie sie zum Beispiel in der Nähe von Elektromotoren vorkommen, aus.

Welche Kassetten verwendet werden sollen

Der ATARI-Programmrecorder 410 kann nur Audiokassetten verarbeiten – keine Digitalkassetten. Mit guten oder sehr guten Normalkassetten sind sie auf jeden Fall gut bedient. Sie sollten jedoch keine Billigkassetten verwenden. Sie neigen dazu, sich nach einer gewissen Zeit zu verklemmen, was gewöhnlich dazu führt, daß Sie Ihre Programme nicht mehr laden können.
Die meisten Programme belegen nicht sehr viel Platz auf der Kassette. Aus diesem Grund sind kurze Bänder meistens praktischer als lange.

Kennzeichnung von Kassetten

Sie sollten jede Kassette mit einem Etikett versehen, auf dem vermerkt ist, was auf dieser Kassette gespeichert ist. Dadurch vermeiden Sie, daß Sie, wenn Sie ein bestimmtes Programm brauchen, eine Kassette nach der anderen mühsam durchsuchen müssen.

Kassetten gegen Beschreiben schützen

An der Rückseite einer jeden Kassette befinden sich zwei Aussparungen (siehe Bild 2-10). Der ATARI-Programmrecorder 410 kann feststellen, ob diese Aussparungen vorhanden oder verdeckt sind. Wenn sie vorhanden sind, kann die Kassette nicht beschrieben werden. Bei neuen Kassetten sind die Aussparungen natürlich verdeckt, damit sie beschrieben werden können. Um eine Kassette gegen Beschreiben zu schützen, brechen Sie die Kunststoffnasen, die die Aussparungen verdecken, einfach heraus. Wenn Sie eine solche Kassette dann später doch noch einmal beschreiben wollen, überkleben Sie die Aussparungen einfach mit einem Stück Klebeband.
Jede Kassette hat zwei Seiten und zwei Aussparungen. Mit der einen Aussparung kann die erste und mit der anderen Aussparung die zweite Seite gegen Beschreiben geschützt werden. Wenn Sie nun wissen wollen, welche Aussparung Sie herausbrechen müssen,

halten Sie die Kassette so, daß das Band zu Ihnen zeigt und die Seite, die Sie schützen wollen, oben liegt. Dann müssen Sie die linke Aussparung herausbrechen.

Bild 2-10: Die Aussparung zum Schutz einer Kassette gegen Beschreiben

Die Bedienung der ATARI-Diskettenstation 810

Wenn an Ihren ATARI-Computer eine oder mehrere Diskettenstationen angeschlossen sind, können Sie Programme und Daten von Disketten an Stelle von Kassetten laden oder darauf ablegen.

Welche Disketten verwendet werden sollten

Die ATARI-Diskettenstation 810 kann sowohl soft- als auch hard-sektorierte 5 1/4-Zoll-Disketten verarbeiten. Soft-sektorierte Disketten sind jedoch vorzuziehen. Wenn Sie eine allgemein bekannte Diskettenmarke verwenden, werden bestimmt keine Probleme auftreten.

Der Umgang mit Disketten

Disketten müssen sehr sorgfältig behandelt werden. Sie sind sehr viel empfindlicher als Kassetten. Knicken Sie niemals eine Diskette und berühren Sie auf keinen Fall die Oberfläche der Diskette selbst (der Teil, der innerhalb der Löcher der Diskettenhülle zu sehen ist). Wenden Sie beim Einlegen einer Diskette in die Diskettenstation niemals Gewalt an. Stecken Sie die Diskette nach jeder Benutzung in die Schutzhülle zurück. Lagern Sie sie nicht zu heiß und setzen Sie sie niemals dem direkten Sonnenlicht oder magnetischen Feldern, wie sie zum Beispiel in der Nähe von Elektromotoren vorkommen, aus. Besonders sorgfältig sollten Sie die mit "Disk File Manager Master Copy" oder "Disk File Manager II Master Copy" bezeichnete Diskette behandeln.

Disketten gegen Beschreiben schützen

Fast alle Disketten haben an der rechten Seite eine kleine Aussparung (siehe Bild 2-11).

Eine Diskette kann nur dann beschrieben werden, wenn diese Aussparung vorhanden ist. Um eine Diskette gegen Beschreiben zu schützen, brauchen Sie diese Aussparung nur mit einem Stück Klebeband zu überkleben.

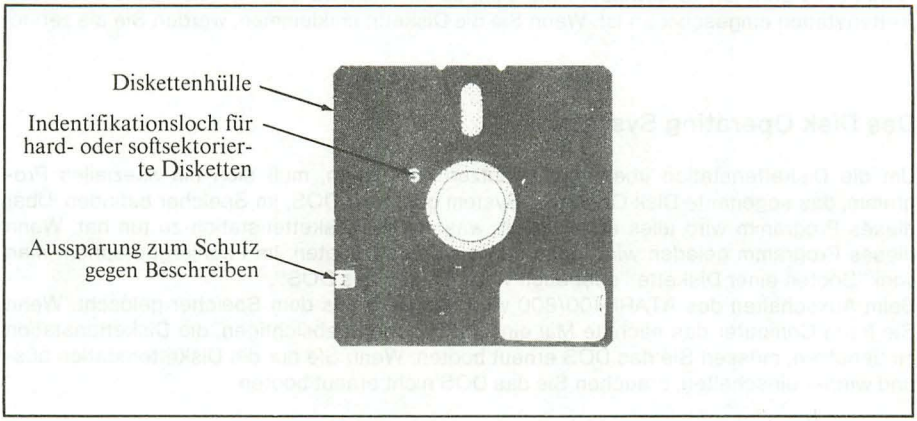


Bild 2-11: Die Aussparung zum Schutz einer Diskette gegen Beschreiben

Einlegen einer Diskette in die Diskettenstation

Bild 2-12 können Sie entnehmen, wie eine Diskette ordnungsgemäß in die Diskettenstation eingelegt wird.Öffnen Sie zunächst die Tür der Diskettenstation. Dann nehmen Sie die Diskette (am besten mit dem Daumen und dem Zeigefinger) und schieben sie sorgfältig bis

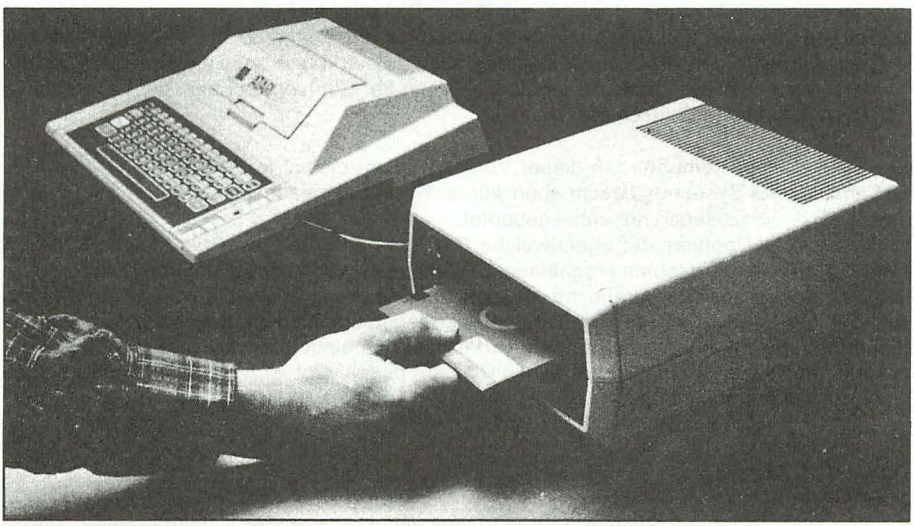


Bild 2-12: Einlegen einer Diskette in die Diskettenstation

zum Anschlag in die Diskettenstation ein. Dabei darf so gut wie gar kein Widerstand spürbar sein. Wenn dem nicht so ist, nehmen Sie die Diskette wieder heraus und versuchen Sie es noch einmal. Halten Sie die Diskette möglichst waagrecht. Nachdem Sie sie eingeschoben haben, schließen Sie die Tür der Diskettenstation wieder. Wenn der Widerstand dabei allzu groß ist, vergewissern Sie sich, ob die Diskette auch wirklich ganz in die Diskettenstation eingeschoben ist. Wenn Sie die Diskette einklemmen, werden Sie sie zerstören.

Das Disk Operating System

Um die Diskettenstation überhaupt benutzen zu können, muß sich ein spezielles Programm, das sogenante Disk Operating System oder kurz DOS, im Speicher befinden. Über dieses Programm wird alles abgewickelt, was mit der Diskettenstation zu tun hat. Wenn dieses Programm geladen wird, dann nennt man das Booten. Im Fachjargon spricht man vom "Booten einer Diskette" oder auch vom "Booten des DOS".
Beim Ausschalten des ATARI 400/800 wird das DOS aus dem Speicher gelöscht. Wenn Sie Ihren Computer das nächste Mal einschalten und beabsichtigen, die Diskettenstation zu benutzen, müssen Sie das DOS erneut booten. Wenn Sie nur die Diskettenstation aus- und wieder einschalten, brauchen Sie das DOS nicht erneut booten.

Booten des Disk Operating Systems

Um das Disk Operating System zu booten, müssen Sie folgendermaßen vorgehen:

1. Schalten Sie die Diskettenstation Nummer 1 ein. Falls Sie mehrere Diskettenstationen besitzen, müssen Sie die heraussuchen, bei der sich die hinter einem Loch an der Rückseite befindlichen Schalter (sowohl der weiße als auch der schwarze) in der linken Schalterposition befinden (siehe Bild 2-4). Bei einem System mit mehreren Diskettenstationen ist das die Diskettenstation Nummer 1.
2. Legen Sie in diese Diskettenstation eine Diskette ein, auf der ein Disk Operating System gespeichert ist. Dies ist zum Beispiel bei der mit "Disk File Manager Master Copy" oder "Disk File Manager Copy" beschrifteten Diskette der Fall.
3. Schalten Sie nun den Computer selbst aus (falls er eingeschaltet war) und wieder ein. Daraufhin wird der Motor der Diskettenstation anlaufen und das DOS von der Diskette in den Speicher des Computers geladen. Wenn der Bootvorgang beendet ist, erscheint "READY" auf dem Bildschirm.

Sie haben sicher bemerkt, daß dieser Vorgang schon einmal in dem Abschnitt über das Einschalten des Systemes beschrieben wurde. Wenn Sie dieser Beschreibung folgen, wird das DOS also sozusagen nebenbei gebootet.
Wenn bei dem Bootvorgang irgendwelche Probleme auftreten, wird die Meldung "BOOT ERROR" auf dem Bildschirm erscheinen. Es kann auch vorkommen, daß die Diskettenstation seltsam schabende Geräusche von sich gibt. Die Ursache eines solchen Fehlers kann zum Beispiel darin liegen, daß die Tür der Diskettenstation nicht geschlossen wurde, keine Diskette eingelegt wurde, die Diskette verkehrt herum eingelegt wurde, kein DOS auf der eingelegten Diskette gespeichert ist, die Diskette beschädigt ist oder die Diskettenstation nicht einwandfrei funktioniert.

Das DOS-Menue

Zu dem Disk Operating System gehören unter anderem mehrere Hilfsprogramme. Einige dieser Hilfsprogramme sind nur für die Anwender gedacht, die auch selbst programmieren.

Andere Hilfsprogramme wiederum wird mit Sicherheit jeder Anwender, der eine Diskettenstation besitzt, irgendwann einmal benutzen müssen. Um diese Hilfsprogramme benutzen zu können, müssen Sie zunächst das DOS booten. Danach geben Sie, ohne die eingelegte Diskette herauszunehmen, über die Tastatur den folgenden Befehl ein:

```
DOS
```

Danach drücken Sie die RETURN-Taste. Bis auf einige eventuell vorhandene kleine Unterschiede müßte auf Ihrem Bildschirm nun dasselbe zu sehen sein wie in Bild 2-13. Diese Aufstellung wird DOS-Menue genannt.

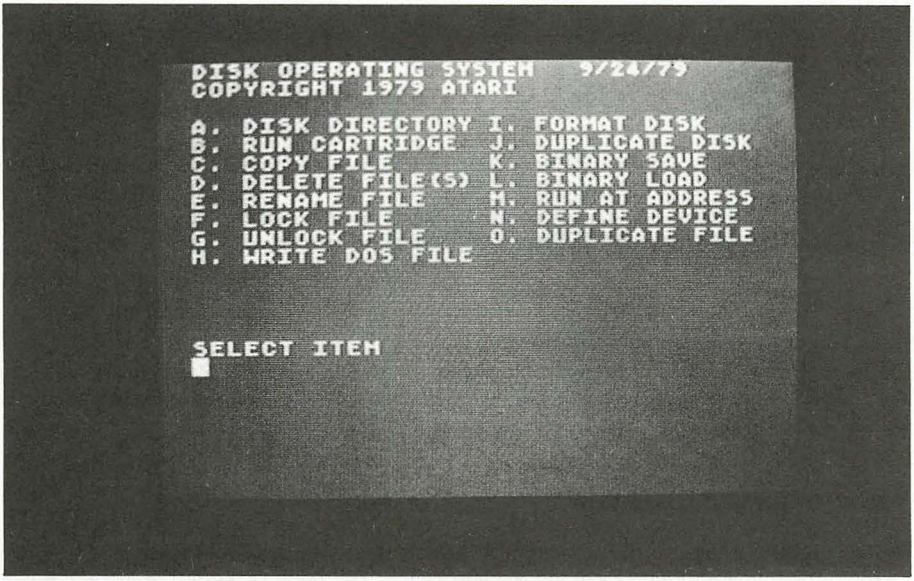


Bild 2-13: Das DOS-Menue

Es gibt zwei verschiedene Versionen des Disk Operating Systems. Die beiden Menues sind zwar nicht völlig gleich, aber sie unterscheiden sich nur unwesentlich. Wenn Sie nicht selber Programme erstellen wollen, brauchen Sie nur die Funktionen A, D, E, I, J, L und O des DOS-Menues. Diese Funktionen sind bei beiden DOS-Versionen dieselben. Bei der einen DOS-Version erscheint das Disk Operating System sofort nach der Eingabe des DOS-Befehls auf dem Bildschirm. Bei der anderen DOS-Version muß der Computer erst noch einmal auf die Diskette zugreifen. Hier kann es maximal 30 Sekunden dauern, bis das DOS-Menue auf dem Bildschirm erscheint.
Achtung: Bei Benutzung des DOS-Befehls wird ein eventuell im Speicher des ATARI-Computers befindliches Programm unter Umständen gelöscht. Aus diesem Grund sollten Sie den DOS-Befehl nicht verwenden, wenn Sie das gerade im Speicher befindliche Programm noch weiter benutzen wollen.

Die Directory einer Diskette

Nachdem Sie eine Diskette gebootet haben, möchten Sie als nächstes wahrscheinlich ger-

ne wissen, welche Programme auf dieser Diskette gespeichert sind. Dazu rufen Sie zunächst (wie oben beschrieben) das DOS-Menue auf. Danach wählen Sie Funktion A des DOS-Menues, indem Sie ein A eingeben und die RETURN-Taste drücken. Daraufhin wird am unteren Ende des Bildschirmes die folgende Meldung ausgegeben:

```
DIRECTORY--SEARCH SPEC,LIST FILE
```

Wenn Sie die RETURN-Taste drücken, werden die Namen aller Programme, die auf dieser Diskette gespeichert sind, auf dem Bildschirm ausgegeben. Wenn nicht alle Programmnamen auf dem Bildschirm Platz finden, weil die Directory (das Inhaltsverzeichnis der Diskette) zu lang ist, versuchen Sie das Ganz noch einmal und unterbrechen Sie dieses Mal die Bildschirmausgabe an einer beliebigen Stelle, indem Sie CTRL-1 drücken. Wenn die Bildschirmausgabe fortgesetzt werden soll, drücken Sie noch einmal CTRL-1.
Um die Directory auf dem Drucker auszudrucken, geben Sie nach der Wahl von Funktion A des DOS-Menues ein Komma, ein P und einen Doppelpunkt ein und drücken Sie erst dann die RETURN-Taste:

```
,P:
```

Wenn Sie mehrere Diskettenstationen besitzen und das Inhaltsverzeichnis einer Diskette ausgegeben werden soll, die sich nicht in der Diskettenstation Nummer 1 befindet, müssen Sie nach der Wahl von Funktion A des DOS-Menues ein D, gefolgt von der Nummer der gewünschten Diskettenstation und einen Doppelpunkt eingeben:

```
D2:
```

Danach drücken Sie die RETURN-Taste.
Es besteht unter anderem auch die Möglichkeit, nur einen Teil der Programmnamen, die auf einer Diskette gespeichert sind, auf dem Bildschirm darzustellen. Eine ausführliche Beschreibung von Funktion A des DOS-Menues finden Sie in Kapitel 7.

Vorbereiten von Disketten

Bei einigen Programmen, die auf Ihrem ATARI-Computer laufen (zum Beispiel bei einem Textverarbeitungsprogramm) werden Sie von Zeit zu Zeit neue Disketten brauchen. Bevor Sie eine neue Diskette benutzen können, müssen Sie sie formatieren. Mit Hilfe der Formatierung wird die Diskette für die weitere Benutzung vorbereitet. Wenn bei dem Programm, für das Sie eine neue Diskette verwenden wollen, gesondert beschrieben ist, wie eine Diskette formatiert werden soll, dann folgen Sie auf jeden Fall dieser Beschreibung. Ansonsten können Sie folgendermaßen vorgehen:
Nachdem Sie das DOS-Menue aufgerufen haben, legen Sie die zu formatierende Diskette in eine Diskettenstation ein und wählen Funktion I des DOS-Menues. Daraufhin wird am unteren Ende des Bidschirmes die folgende Meldung ausgegeben:

```
WHICH DRIVE TO FORMAT?
```

Geben Sie nun ein D, gefolgt von der Nummer der Diskettenstation, in der sich die zu formatierende Diskette befindet, ein (D1, D2, D3 oder D4) und drücken Sie die RETURN-Taste.
Diese Eingabe müssen Sie dann noch einmal bestätigen, indem Sie Y eingeben. Bei einer anderen Eingabe wird Funktion I des DOS-Menues abgebrochen. Nachdem Sie Y eingegeben haben, wird die in der angegebenen Diskettenstation eingelegte Diskette formatiert. Das dauert etwa eine Minute.
Versehen Sie die neu formatierte Diskette am besten gleich mit einem Etikett.

Achtung: Bei der Formatierung einer Diskette wird alles, was auf dieser Diskette eventuell gespeichert ist, gelöscht.

Duplizieren von Disketten

Oft werden von wichtigen Disketten sicherheitshalber Kopien, sogenannte Backups angefertigt, die nur in dem Fall benutzt werden, wenn die Originaldiskette durch irgeneinen Umstand zerstört worden ist. Dazu dient Funktion J des DOS-Menues. Damit können Sie auch dann Disketten kopieren, wenn Sie nur eine Diskettenstation haben. Nach der Wahl von Funktion J des DOS-Menues erscheint am unteren Bildschirmrand die folgende Meldung:

```
DUP DISK--SOURCE,DEST DRIVES?
```

Bevor Sie weitermachen, sollten Sie an der Originaldiskette einen Schreibschutz anbringen. Das ist eine simple Vorsichtsmaßnahme, mit der Sie sich viel Ärger ersparen, falls Sie einmal einen Eingabefehler machen sollten.
Geben Sie nun nacheinander die Nummer der Diskettenstation, in die Sie die Originaldiskette einlegen wollen (Source = Quelle), ein Komma und die Nummer der Diskettenstation, in die Sie die zukünftige Backup-Diskette einlegen wollen (Destination = Ziel) an. Das sieht dann zum Beispiel so aus:

```
D1,D1
```

Falls Sie für Quelle und Ziel dieselbe Diskettenstation angegeben haben, erscheint daraufhin die folgende Meldung auf dem Bildschirm:

```
TYPE "Y" IF OK TO USE PROGRAM AREA?
```

Achtung: Wenn Sie diese Frage bejahen, indem Sie Y eingeben, beginnt der Kopiervorgang. Dabei kann ein eventuell gerade im Speicher befindliches Programm gelöscht werden. Wenn Sie dieses Programm noch weiter benutzen wollen, dürfen Sie hier kein Y eingeben.

Um den Kopiervorgang zu starten, geben Sie nun Y ein und drücken die RETURN-Taste. Bei jeder anderen Eingabe wird das Kopierprogramm abgebrochen. Daraufhin erscheint eine Meldung auf dem Bildschirm, mit deren Hilfe Sie dazu aufgefordert werden, die Quell- und Zieldiskette einzulegen. Wenn Sie für die Quell- und die Zieldiskette dieselbe Diskettenstation verwenden, müssen Sie zunächst die Quelldiskette und dann, nachdem der Computer einen Teil der Quelldiskette gelesen hat, die Zieldiskette einlegen. Wenn dann der vorher gelesene Teil der Quelldiskette auf die Zieldiskette geschrieben worden ist, müssen Sie wieder die Quelldiskette einlegen. Dieser Wechsel kann sich durchaus noch mehrere Male wiederholen. Dazu erscheint jedesmal eine entsprechende Meldung auf dem Bildschirm. Nach jedem Diskettenwechsel müssen Sie die RETURN-Taste drücken. Damit teilen Sie dem Computer mit, daß die gewünschte Diskette eingelegt und die Tür der Diskettenstation geschlossen ist.
Falls Sie einmal die Quell- und die Zieldiskette verwechseln sollten, dann wird, sofern Sie die Quelldiskette vorher schreibgeschützt haben, eine Fehlermeldung auf dem Bildschirm erscheinen. In diesem Fall müssen Sie den Kopiervorgang noch einmal von vorne beginnen. Wenn die Quelldiskette nicht schreibgeschützt ist, kann sie, wenn Sie sie an Stelle der Zieldiskette verwenden, unter Umständen zerstört werden.
Unter bestimmten Bedingungen kann es vorkommen, daß Sie eine auf diese Weise angefertigte Kopie einer Diskette nicht booten können. Um diesen Fehler zu beheben, booten Sie zunächst das DOS von einer anderen Diskette, rufen das DOS-Menue auf und wählen

dann Funktion H des DOS-Menues. Daraufhin wird die folgende Meldung auf dem Bildschrim erscheinen:

```
DRIVE TO WRITE DOS FILES TO?
```

Legen Sie die Diskette, die Sie nicht booten können, in eine Diskettenstation ein, geben Sie die Nummer dieser Diskettenstation an (D1, D2, D3 oder D4) und drücken Sie die RETURN-Taste. Daraufhin wird eine andere Meldung auf dem Bildschirm erscheinen:

```
TYPE "Y" TO WRITE DOS TO DRIVE 1?
```

Wenn Sie diese Frage bejahen, indem Sie Y eingeben, wird eine Kopie des Disk Operating Systems auf der in der angegebenen Diskettenstation eingelegten Diskette gespeichert.

Duplizieren von Programmen

Mit Hilfe von Funktion O des DOS-Menues können Sie ein auf einer Diskette gespeichertes Programm auf eine andere Diskette kopieren. Nach der Wahl dieser Funktion erscheint die folgende Meldung auf dem Bildschirm:

```
NAME OF FILE TO MOVE?
```

Geben Sie nun den Namen des Programmes ein, das Sie kopieren möchten, zum Beispiel:

```
TEXT.BAS
```

Danach drücken Sie die RETURN-Taste. Es wird davon ausgegangen, daß das angegebene Programm auf der in Diskettenstation Nummer 1 eingelegten Diskette gespeichert ist. Aus diesem Grund brauchen sie hier keine Diskettenstation angeben. Als nächstes erscheint dann die folgende Meldung auf dem Bildschirm:

```
TYPE "Y" IF OK TO USE PROGRAM AREA?
  CAUTION: A "Y" INVALIDATES MEM.SAV
```

Achtung: Wenn Sie diese Frage bejahen, indem Sie Y eingeben, wird bei dem Kopiervorgang ein eventuell gerade im Speicher befindliches Programm unter Umständen gelöscht. Wenn Sie dieses Programm noch weiter benutzen wollen, dürfen Sie hier kein Y eingeben.

Im Gegensatz zu dem oben beschriebenen Programm zum Kopieren von Disketten findet der Kopiervorgang hier immer statt, ganz gleich, wie Sie diese Frage beantworten. Als nächstes werden Sie dann dazu aufgefordert, zuerst die Quell- und dann die Zieldiskette einzulegen. Unter Umständen müssen Sie die Disketten auch mehrmals wechseln. In diesem Fall liest der Computer einen Teil des Programmes von der Quelldiskette, speichert diesen Teil auf der Zieldiskette, lädt ein weiteres Programmstück von der Quelldiskette usw., bis das ganze Programm kopiert ist. Wenn das Programm nicht allzu lang ist und Sie die obige Frage mit Y beantwortet haben, brauchen Sie die Quell- und die Zieldiskette normalerweise nur einmal anzulegen. Jedesmal, wenn Sie die Diskette wechseln müssen, erscheint eine entsprechende Meldung auf dem Bildschirm. Nach jedem Diskettenwechsel müssen Sie die RETURN-Taste drücken. Damit teilen Sie dem Computer mit, daß die gewünschte Diskette eingelegt und die Tür der Diskettenstation geschlossen ist.
Mit Hilfe von Funktion C des DOS-Menues können Sie ein Programm kopieren und dabei die Quell- und Zieldiskette in zwei verschiedene Diskettenstationen einlegen. Eine vollständige Beschreibung von Funktion C des DOS-Menues finden Sie in Kapitel 7.

Löschen eines Programmes

Um ein Programm von einer Diskette zu löschen, wählen Sie Funktion D des DOS-Menues. Daraufhin erscheint die folgende Meldung auf dem Bildschirm:

```
DELETE FILE SPEC
```

Geben Sie nun die Nummer der Diskettenstation an, in der sich die Diskette befindet, auf der das zu löschende Programm gespeichert ist, gefolgt von einem Doppelpunkt und dem Namen des Programmes, das gelöscht werden soll. Das sieht dann zum Beispiel so aus:

```
D2:HANGMAN
```

Danach drücken Sie die RETURN-Taste. Wenn sich die fragliche Diskette in der Diskettenstation Nummer 1 befindet, können Sie die Nummer der Diskettenstation und den Doppelpunkt weglassen.

Umbenennen eines Programmes

Sie können Ihren Programmen jeden beliebigen Programmnamen geben. Einige Einschränkungen gibt es jedoch: Erstens dürfen niemals zwei Programme auf ein und derselben Diskette denselben Namen haben. Zweitens dürfen die Programmnamen nicht länger als acht Zeichen sein. Für einen Programmnamen dürfen nur die Großbuchstaben von A bis Z und die Ziffern von 0 bis 9 verwendet werden. Das erste Zeichen eines Filenamens muß ein Großbuchstabe sein. Wenn Sie wollen, können Sie hinter einem Programmnamen einen Punkt und noch einmal maximal drei Zeichen angeben. Man nennt das den Zusatz eines Filenamens. Der Zusatz .SYS ist reserviert. Näheres darüber erfahren Sie in Kapitel 7.
Wenn Sie ein Programm umbenennen wollen, wählen Sie Funktion E des DOS-Menues. Danach erscheint die folgende Meldung auf dem Bildschirm:

```
RENAME - GIVE OLD NAME, NEW
```

Geben Sie zuerst den Namen an, den das Programm im Moment trägt, dann ein Komma und danach den Namen, den das Programm zukünftig tragen soll. Das sieht dann zum Beispiel so aus:

```
TEXT10.TXT,EINLADNG
```

Danach drücken Sie die RETURN-Taste. Es wird davon ausgegangen, daß sich das angegebene Programm auf der in Diskettenstation Nummer 1 eingelegten Diskette befindet Deshalb brauchen Sie hier keine Diskettenstation angeben.

Laden und Starten von Programmen

Es gibt sehr viele Programme für den ATARI-Computer, die sie fertig kaufen können. Manche sind auf Kassette und andere auf Diskette gespeichert. Bevor Sie ein solches Programm benutzen können, müssen Sie es von der Kassette oder Diskette in den Hauptspeicher des Computers übertragen. Man sagt, das Programm wird geladen. Nachdem ein Programm geladen worden ist, kann es gestartet werden.

Laden eines auf Kassette gespeicherten Programmes

Es gibt beim ATARI-Computer drei verschiedene Befehle, mit deren Hilfe ein auf Kassette

gespeichertes Programm geladen werden kann. Diese drei Befehle sind untereinander nicht austauschbar. Welcher Befehl verwendet werden muß, hängt davon ab, auf welche Art das zu ladende Programm gespeichert wurde. Wenn Sie nicht wissen, welchen Befehl Sie benutzen müssen, können Sie die verschiedenen Möglichkeiten der Reihe nach durchprobieren, bis das Programm fehlerfrei geladen wird. Die drei Befehle lauten CLOAD, ENTER "C:" und LOAD "C:".
Um ein auf Kassette gespeichertes Programm zu laden, gehen sie folgendermaßen vor:

1. Positionieren Sie das Band an der Stelle, an der das zu ladende Programm gespeichert ist. Dazu spulen sie das Band zunächst ganz zurück und setzen das Zählwerk des Programmrecorders auf Null. Wenn das zu ladende Programm am Anfang der eingelegten Kassette gespeichert ist, können Sie danach zu Schritt 2 übergehen. Ansonsten müssen Sie das Band mit Hilfe des Zählwerks und der ADVANCE-Taste soweit vorspulen, bis die Position erreicht ist, an der das gewünschte Programm gespeichert ist. Wenn Sie nicht wissen, bei welchem Zählwerksstand das ist, müssen Sie alle auf der Kassette gespeicherten Programme nacheinander laden, bis das gewünschte Programm erreicht worden ist. In diesem Fall müssen Sie die folgenden Schritte für jedes Programm, das Sie laden, wiederholen.
2. Geben Sie über die Tastatur einen der drei Befehle zum Laden eines auf Kassette gespeicherten Programmes (CLOAD, ENTER "C:" oder LOAD "C:") ein. Nachdem Sie die RETURN-Taste gedrückt haben, wird ein Kontrollton ertönen.
3. Drücken Sie am Programmrecorder die PLAY-Taste. Der ATARI-Computer kann nicht feststellen, ob Sie das getan haben. Wenn Sie es nicht tun, wird bei Schritt 4 ein Fehler auftreten.
4. Drücken Sie nun noch einmal die RETURN-Taste. Daraufhin wird der Motor des Programmrecorders anlaufen. Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie nach einer kurzen Pause mehrere schnelle Tonfolgen hören: Das Programm wird geladen.

Wenn während des Ladevorganges ein Fehler auftritt, haben sie höchstwahrscheinlich den falschen Ladebefehl benutzt. Versuchen Sie noch einmal das Programm zu laden und verwenden Sie dieses Mal einen anderen Ladebefehl. Falls Sie auch mit den beiden anderen Ladebefehlen keinen Erfolg haben, ist mit ziemlicher Sicherheit die Kassette leer, zerstört, beschädigt oder verkehrt herum eingelegt.

Laden eines auf Diskette gespeicherten Programmes

Einige Programme werden beim Booten des DOS automatisch geladen und gestartet. In diesem Fall müssen Sie beim Einschalten des Computers (siehe Anfang dieses Kapitels) nur die richtige Diskette einlegen.
Bevor ein auf Diskette gespeichertes Programm geladen werden kann, muß das DOS gebootet werden. Danach können Sie mit einem der beiden folgenden Befehle ein auf Diskette gespeichertes Programm laden: ENTER "Programm" oder LOAD "Programm". Für "Programm" wird dabei die Nummer einer Diskettenstation, gefolgt von einem Doppelpunkt und einem Programmnamen eingesetzt. Das sieht dann zum Beispiel so aus:

```
LOAD "D1:VERWALT.BAS"
```

Wenn Sie ein Programm von der in der Diskettenstation Nummer 1 eingelegten Diskette laden wollen (also auch im obigen Beispiel), dann können Sie bei der Bezeichnung der Diskettenstation die 1 weglassen.

Starten eines Programmes

Wenn Sie ein von Kassette oder Diskette geladenes Programm starten wollen, müssen Sie RUN eingeben und die RETURN-Taste drücken. Das Programm übernimmt daraufhin die Kontrolle über den Computer, auch über die Tastatur und den Bildschirm. Wenn Sie das Programm unterbrechen und selbst wieder die Kontrolle übernehmen wollen, drücken Sie die BREAK-Taste. Bei den meisten Programmen funktioniert das. Falls es nicht funktionieren sollte, müssen Sie in der Bedienungsanleitung des Programmes, das Sie gerade benutzen, nachsehen, wie das Programm unterbrochen werden kann. In Notfällen können Sie auch die SYSTEM RESET-Taste drücken oder den Computer aus- und wieder einschalten. In beiden Fällen müssen Sie allerdings das Programm erneut starten und eventuell (beim Ausschalten des Computers auf jeden Fall) sogar erneut laden.
Es gibt einen Befehl, mit dessen Hilfe ein auf Kassette gespeichertes Programm geladen und danach automatisch gestartet wird. Es handelt sich dabei um den RUN "C:"-Befehl. Sie können ihn an Stelle des LOAD "C:"-Befehls verwenden. Es ist nicht möglich, damit ein Programm zu laden und zu starten, das normalerweise mit Hilfe des CLOAD- oder des ENTER "C:"-Befehls geladen wird.
Einen entsprechenden Befehl gibt es auch für Programme, die auf Diskette gespeichert sind. Es handelt sich dabei um den RUN "Programm"-Befehl. Sie können ihn an Stelle des LOAD "Programm"-Befehls verwenden. Es ist nicht möglich, damit ein Programm zu laden und zu starten, das normalerweise mit Hilfe des ENTER "Programm"-Befehls geladen wird.

Farbeinstellung des Fernsehgerätes

Mit dem ATARI-Computer können vollwertige Farbgrafiken erstellt werden. Um solche Farbgrafiken optimal darstellen zu können, muß Ihr Fernsehgerät oder Monitor richtig eingestellt sein. Es ist sicher nicht falsch, wenn Sie dieselbe Einstellung wie beim fernsehen verwenden. Wenn Sie wollen, können Sie die Einstellung von Kontrast, Helligkeit und Farbsättigung noch etwas korrigieren, um die Bildqualität zu optimieren.

Benutzung der Regler

An der Vorderseite des ATARI 400/800 befinden sich vier Anschlüsse für verschiedene

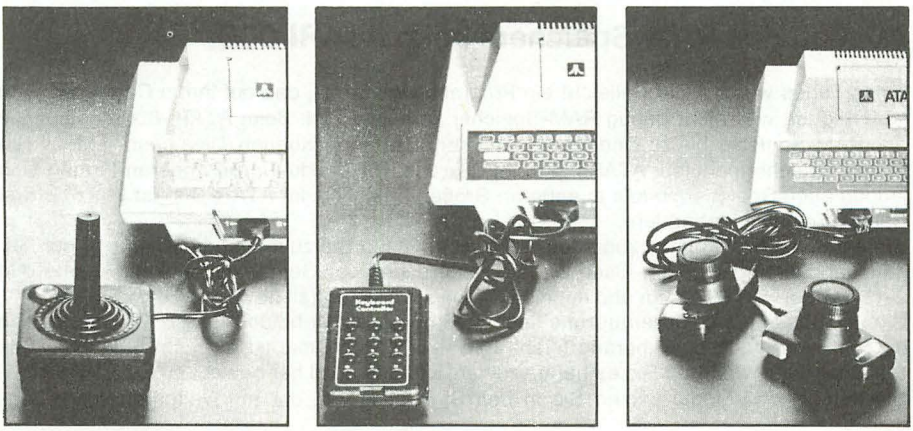


Bild 2-14: Die verschiedenen Regler

Regler (siehe Bild 2-14). Welchen Anschluß Sie verwenden müssen, sollte in der Bedienungsanleitung des entsprechenden Programmes vermerkt sein. Falls dem nicht so ist, probieren Sie, beginnend mit Anschluß 1 auf der linken Seite, nacheinander alle Anschlüsse durch.
Die Knöpfe der Drehregler können über fast eine ganze Umdrehung gedreht werden. Es wird allerdings nicht der ganze Drehbereich genutzt. Wenn der Knopf am rechten Anschlag steht und dann langsam im Gegenuhrzeigersinn an den linken Anschlag gedreht wird, dann ist nur bei den ersten zwei Dritteln dieser Drehung eine Wirkung feststellbar. Das letzte Drittel ist ungenutzt.
Die Steuerknüppel sind zwar sehr stabil, aber sie können beschädigt werden, wenn über eine längere Zeit eine große Kraft auf sie einwirkt. Die Steuerknüppel sprechen immer gleich schnell an, egal ob sie nur leicht oder mit großem Kraftaufwand betätigt werden. Eine etwas gemäßigte Behandlung werden sie Ihnen durch eine lange Lebensdauer danken.

Benutzung des ATARI-Interfacemoduls 850

Wenn Sie ein ATARI-Interfacemodul 850 haben, werden Sie wahrscheinlich auch einen ATARI-Drucker 825 oder ein Gerät, das an eine der seriellen Schnittstellen angeschlossen wird, besitzen. Falls an Ihr Interfacemodul nur der ATARI-Drucker 825 angeschlossen ist, dann können Sie es ohne weiteres ausschalten, wenn Sie gerade nicht drucken. Wenn hingegen irgendein Gerät an die seriellen Schnittstellen angeschlossen ist, dann muß das Interfacemodul eingeschaltet bleiben.

Benutzung eines Druckers

Ganz gleich, welchen Drucker Sie benutzen: er muß nur dann eingeschaltet sein, wenn gedruckt werden soll. Etwas Vorsicht ist allerdings angebracht, denn es kann unter Umständen ein Fehler auftreten, wenn in einem Programm der Drucker angesprochen wird und dieser gerade nicht eingeschaltet ist.
Beim ATARI-Drucker gibt es einen Schalter mit den Schalterstellungen "ONLINE" und "LOCAL". Wenn Sie etwas drucken wollen, dann muß dieser Schalter in der "ONLINE"-Stellung stehen. In der "LOCAL"-Stellung können Sie das Papier mit Hilfe des REV/FWD-Schalters vor und zurück transportieren.

Ausbau des RAM-Speichers beim ATARI 800

Eines Tages werden Sie vielleicht ein Programm erwerben, das auf Ihrem Computer nicht lauffähig ist, weil nicht genug RAM-Speicher vorhanden ist. Beim ATARI 800 können Sie den RAM-Speicher bis zu einer gewissen Grenze selbst ausbauen. Dies geschieht mit Hilfe von Speichermodulen. ATARI vertreibt 8K- und 16K-Module. Von anderen Firmen sind jedoch auch Speichermodule in anderen Größen erhältlich. Im ATARI 800 ist Platz für drei Speichermodule vorhanden.
Um auf diese Steckplätze zugreifen zu können, öffnen Sie zunächst die Klappe in der Gehäuseabdeckung. Danach lösen Sie die Halterungen der Gehäuseabdeckung (siehe Bild 2-15), heben Sie etwas an und nehmen Sie nach vorne ab (siehe Bild 2-16).
Um ein Speichermodul herauszunehmen, halten Sie es an beiden Enden fest und ziehen es senkrecht nach oben heraus (siehe Bild 2-17). Manchmal ist es recht hilfreich, wenn Sie das Modul, während Sie es herausziehen, leicht hin und her bewegen.
Ein neues RAM-Modul setzen Sie in den Steckplatz ein, der am weitesten vorne liegt.

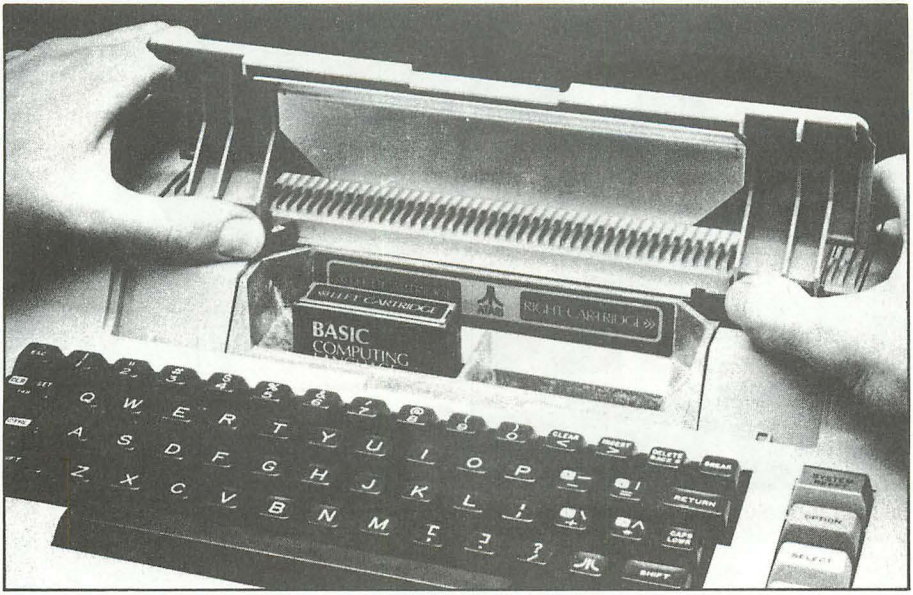


Bild 2-15: Lösen der Halterungen der Gehäuseabdeckungen des ATARI 800

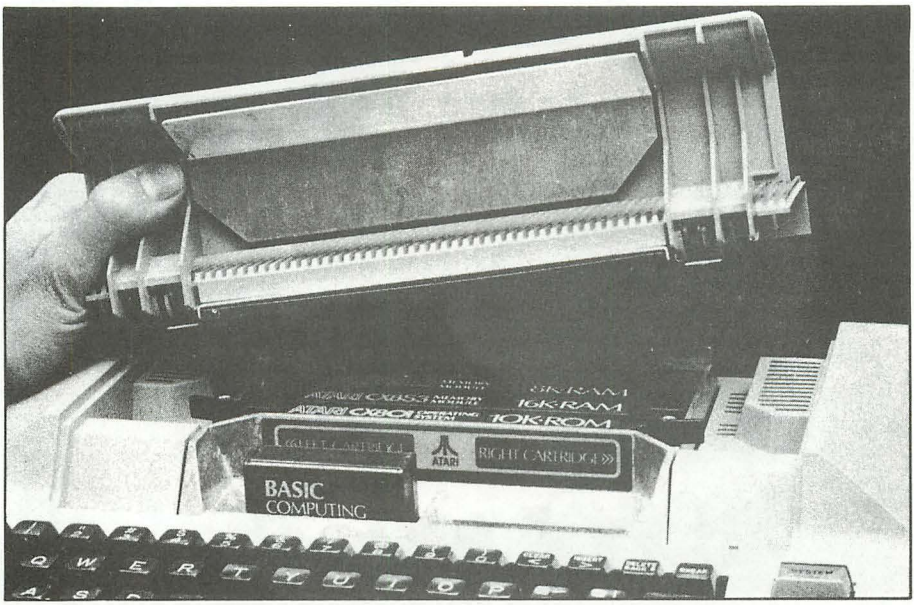


Bild 2-16: Abnehmen der Gehäuseabdeckung des ATARI 800

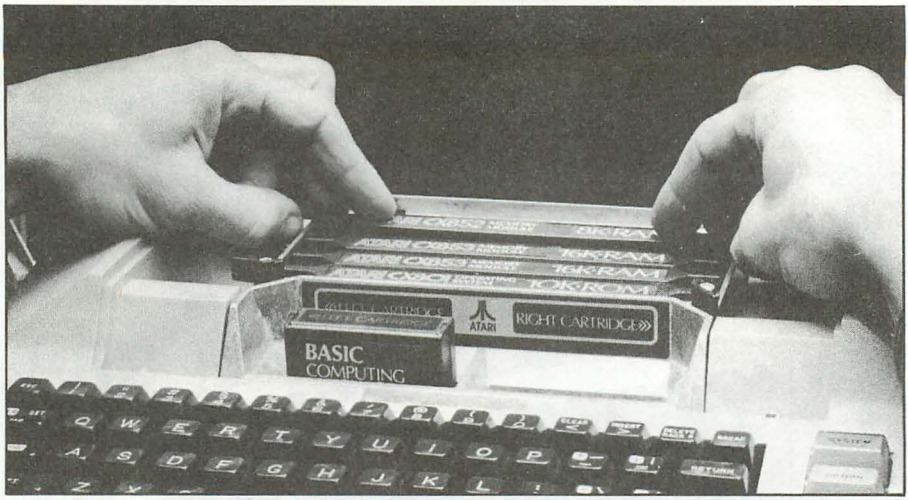


Bild 2-17: Herausnehmen eines Speichermoduls (nur beim ATARI 800)

Bild 2-18: Befestigung der Gehäuseabdeckung des ATARI 800 mit Hilfe der Halterungen

Drücken Sie das Modul an beiden Enden fest in seine Halterung. Die Steckplätze müssen von vorne nach hinten gefüllt werden. Es darf kein weiter vorne oder zwischen zwei Modulen liegender Steckplatz frei bleiben. Wenn Sie sowohl 8K- als auch 16 K-Module verwenden, müssen Sie die 16K-Module in die vorderen Steckplätze einsetzen.
Um die Gehäuseabdeckung wieder aufzusetzen, müssen Sie zunächst die zwei Metallzungen der Gehäuseabdeckung in die entsprechenden Löcher im Chassis des ATARI 800 einführen. Schieben Sie die Abdeckung nach hinten und drücken Sie sie so weit nach unten, bis sie mit dem Gehäuse des ATARI 800 bündig abschließt. Dann befestigen Sie die Gehäuseabdeckung wieder mit Hilfe der beiden Halterungen (siehe Bild 2-18) und schließen die Klappe der Abdeckung.

Was tun, wenn ein Fehler auftritt

Der ATARI-Computer ist zwar sehr leistungsfähig, aber er kann, wie alle anderen Computer auch, nicht kreativ sein. Aus diesem Grund wird, wenn Sie ihm einen Befehl erteilen, der nicht der vorgeschriebenen Form entspricht, ein Fehler auftreten. Die Auswirkungen eines solchen Eingabefehlers können leicht bis verheerend sein.

Die Fehlermeldungen

Wenn Sie bei einer Eingabe einen Fehler gemacht haben, wird, nachdem Sie die RETURN-Taste gedrückt haben, eine Fehlermeldung auf dem Bildschirm erscheinen. Oftmals enthält diese Fehlermeldung einen Hinweis darauf, was Sie falsch gemacht haben. Manchmal ist das jedoch nicht der Fall. Sehr oft besteht die Fehlermeldung nur aus dem Wort ERROR und einer Zahl. Eine ausführliche Beschreibung dieser Fehlermeldungen finden Sie in Teil A des Anhanges.
Wenn im Verlauf eines Programmes eine Fehlermeldung auftritt, sehen Sie in der Bedienungsanleitung des Programmes nach, was Sie falsch gemacht haben.

Korrektur von Eingabefehlern

Bei der Eingabe von Befehlen über die Tastatur werden Sie mit ziemlicher Sicherheit auch einmal einen Fehler machen. Mit Hilfe von einigen der Tasten, die in diesem Kapitel bereits beschrieben wurden, können Sie Eingabefehler, die Sie noch vor dem Drücken der RETURN-Taste bemerken, sehr einfach korrigieren. Es handelt sich dabei um die BACK S-, die ← (CTRL- + -), die → (CTRL-^ -), die TAB-, die BREAK- und die CLEAR- (SHIFT-<-) Taste.

- Mit Hilfe der BACK S-Taste wird der Cursor nach links bewegt. Dabei werden die Zeichen, über die der Cursor hinwegbewegt wird, gelöscht und durch Leerzeichen ersetzt.
- Die ← Taste bewegt den Cursor um eine Stelle nach links. Dabei werden keine Zeichen gelöscht.
- Die → Taste bewegt den Cursor um eine Stelle nach rechts. Genauso wie bei der ← Taste werden auch hier keine Zeichen gelöscht.
- Mit Hilfe der TAB-Taste wird der Cursor an die Stelle bewegt, an der sich der nächste Tabulator-Stop befindet. Auch hier werden keine Zeichen gelöscht.
- Bei Betätigung der BREAK-Taste springt der Cursor an den Anfang der nächsten Zeile. Die Zeile, in der er sich vorher befand, wird ignoriert.
- Mit Hilfe der CLEAR-Taste kann der Bildschirm gelöscht werden. Der Cursor befindet sich anschließend in der oberen linken Ecke des Bildschirmes.

Die Benutzung dieser Tasten soll an Hand eines Beispieles erläutert werden. Nehmen Sie einmal an, Sie wollten den folgenden Befehl eingeben:

```
LOAD "D1:MUSIK"
```

Doch bevor Sie die RETURN-Taste drücken, stellen Sie fest, daß Sie einen Fehler gemacht haben:

```
LOAS "D1:MUSIK"
```

Um diesen Fehler zu korrigieren, gibt es mehrere Möglichkeiten. Eine davon ist, die BREAK-Taste zu drücken und den ganzen Befehl noch einmal einzugeben. Genausogut können Sie den Fehler jedoch auch mit Hilfe der ← Taste oder der BACK S-Taste beheben.
Versuchen Sie es einmal mit der ← Taste. Drücken Sie gleichzeitig die CTRL- und die ←-Taste und lassen Sie die ←-Taste erst dann wieder los, wenn der nach links laufende Cursor das fehlerhafte S erreicht hat. Wenn Sie über das Ziel hinausgeschossen sind, können Sie den Cursor mit Hilfe der → Taste wieder nach rechts bewegen. Wenn Sie, sobald der Cursor über dem fehlerhaften S steht, ein D eingeben, ist der Fehler behoben und Sie können die RETURN-Taste betätigen. Es ist nicht nötig, den Cursor vorher an das Ende der Zeile zu bewegen.

Was tun, wenn versehentlich die BREAK-Taste gedrückt wurde

Früher oder später werden Sie bestimmt einmal versehentlich die BREAK-Taste drücken. Einige Programme sind so aufgebaut, daß die BREAK-Taste überhaupt keinen Einfluß auf den Programmablauf hat. Ansonsten sollten in der Bedienungsanleitung besondere Anweisungen für den Fall, daß sie die BREAK-Taste aus Versehen einmal betätigen, enthalten sein. Aus diesem Grund sollten Sie sich die Bedienungsanleitung eines Programmes sorgfältig durchlesen, bevor Sie mit dem Programm arbeiten, um im Fall des Falles zu wissen, was zu tun ist. Es ist grundsätzlich möglich, ein BASIC-Programm, das durch Drücken der BREAK-Taste unterbrochen wurde, neu zu starten. Davon haben Sie aber bei manchen Anwendungen, zum Beispiel bei einem Buchhaltungsprogramm, ziemlich wenig.
Sobald die BREAK-Taste gedrückt wird, unterbricht der Computer das gerade laufende Programm und übergibt Ihnen die Kontrolle über den Computer. Um Ihnen das mitzuteilen, erscheint die folgende Meldung auf dem Bildschirm:

```
STOPPED AT LINE XXXX
```

wobei für XXXX die Zeilennummer eingesetzt wird, in der das Programm unterbrochen wurde. In manchen Fällen kann das Programm fortgesetzt werden, indem Sie CONT eingeben. Wenn das nicht funktioniert, haben Sie Pech gehabt und müssen das Programm neu starten. Aber bevor Sie RUN eingeben, sollten Sie sicher sein, daß durch einen neuen Programmstart nicht noch größerer Schaden verursacht wird. Sehen Sie deshalb in der Bedienungsanleitung des Programmes nach oder fragen Sie Ihren Händler oder irgendjemand, der das Programm ebenfalls benutzt. Unter Umständen ist die Lösung des Problems gar nicht so einfach.

3
PROGRAMMIEREN IN BASIC

BASIC ist eine Computer-Programmiersprache. Sie besteht aus einer Folge von Anweisungen und Befehlen. Jede Anweisung und jeder Befehl teilt dem Computer mit, eine einfache Aufgabe auszuführen. Um komplizierte Aufgaben zu lösen, gibt man ihm Anweisungen in Form von unterschiedlichen, aufeinander aufbauenden BASIC-Befehlen. Ein Programm ist einfach ausgedrückt eine Folge von Anweisungen. Programmieren heißt, diese Befehle in einem sinnvollen Verfahren, das der Computer verstehen und daraufhin ausführen kann, zusammenzufassen.

In diesem Kapitel lernen Sie, Ihre eigenen BASIC-Programme auf dem ATARI-Computer zu schreiben. Man könnte jetzt anfangen, alle Einzelheiten über jeden BASIC-Befehl nacheinander zu lernen, aber dann würden Sie sicherlich bald aufgeben.

Befehle allein bedeuten nicht viel; es kommt darauf an, sie richtig zu kombinieren. Das Erlernen der einzelnen BASIC-Anweisungen ohne die Zusammenhänge zu verstehen, würde Ihnen bald erscheinen, als lernten Sie eine Menge unsinniger Regeln. Sie erfahren dadurch weder etwas über das eigentliche Programmieren, noch vermittelt es Ihnen Programmierpraxis.

In diesem Kapitel sind alle BASIC-Anweisungen in der logischen Reihenfolge beschrieben und anhand von praktischen Beispielen erklärt. In Kapitel 11 können Sie die exakten Definitionen und Feinheiten der einzelnen Befehle nachlesen. Sie sollten jedoch nicht versuchen, nur anhand dieser Erläuterungen das Programmieren zu lernen.

BASIC-Beginn

Es gibt mindestens drei verschiedene BASIC-Versionen für den ATARI-Computer. Dieses Buch behandelt ausschließlich die Standard Version, die mit dem ATARI 400 und ATARI 800 Computer geliefert wird. Sie befindet sich in dem ROM-Modul, das mit "BASIC-Programmier-Sprache" bezeichnet ist (Nr. CXL4002). Die anderen BASIC-Versionen sind dem Standard ATARI-BASIC ähnlich, weichen jedoch in Details ab.

Installieren des BASIC-ROM-Moduls

Der ATARI-Computer ist sehr vielseitig. Mit ihm können Sie Musik komponieren, Lernprogramme anwenden, sich mit den zahlreichen Computerspielen beschäftigen, sowie vieles andere mehr. Möchten Sie jedoch in ATARI-BASIC programmieren, dann muß das ROM-Modul "BASIC-Programmier-Sprache" eingesetzt sein. Die Instruktionen zum Installieren des Moduls finden Sie in Kapitel 2 (Absatz 2-1).

Anschalten des Computers

In Kapitel 2 finden Sie auch die korrekte Reihenfolge, in der die verschiedenen ATARI-Computer-Komponenten eingeschaltet werden müssen. Es ist wichtig, diese Vorschriften genau zu befolgen, da der Computer nur korrekt mit den angeschlossenen Geräten kommunizieren kann, wenn die Reihenfolge eingehalten wird. BASIC ist betriebsbereit, wenn die Meldung READY auf dem Bildschirm dargestellt ist.

Verlassen der Programmiersprache BASIC

Es gibt zwei Möglichkeiten, die Programmiersprache BASIC zu verlassen. Die eine besteht darin, das BASIC-Modul aus dem Modulschacht zu entfernen. Sollten sie gerade an einem BASIC-Programm arbeiten, so wird dieses gelöscht, da sich der Computer automatisch abschaltet, sobald das Modul herausgenommen wird. Die zweite Möglichkeit ist, den Befehl BYE zu schreiben und RETURN zu drücken. Dadurch wird der Computer in die Betriebsart Bildschirmdarstellung (Memo Pad Mode) versetzt. In dieser Betriebsart schreiben Sie von der Tastatur nur auf den Bildschirm, ohne daß die Eingabe ausgeführt oder programmiert wird. Durch Betätigen der Taste SYSTEM RESET gelangen Sie wieder in die Programmiersprache BASIC.

PRINT

BASIC meldet sich mit der Meldung READY betriebsbereit. Die Anweisungen, die Sie jetzt schreiben, erteilen Sie ihm in der sofort ausführenden Betriebsart. Diese nennt man auch direkte oder Rechner-Betriebsart. Der Computer antwortet sofort auf jede Ihrer Anweisungen. Versuchen Sie es mit diesem Beispiel:

```
PRINT "WIE GEHT ES IHNEN"
```

Vergessen Sie nicht, die RETURN-Taste nach den letzten Anführungsstrichen zu drücken. Der Computer antwortet sofort:

```
WIE GEHT ES IHNEN

READY
```

Vielleicht schreibt der Computer aber auch die Meldung ERROR-, gefolgt von Ihrer Eingabe. Das bedeutet, daß er Ihren Befehl nicht verstanden hat. Wahrscheinlich haben Sie das Wort PRINT falsch geschrieben. Wenn er jedoch nur mit der Zahl 0 antwortet, bedeutet das, daß Sie die ersten Anführungsstriche vergessen haben. In beiden Fällen können Sie Ihre Eingabe einfach wiederholen, wobei Sie jetzt natürlich diese Fehler vermeiden sollten. Computer nehmen es sehr genau mit dem Buchstabieren und der Interpunktion. Schon der kleinste Fehler kann das Programm stoppen oder, was schlimmer wäre, den Computer verlassen, Dinge zu tun, die Sie nicht beabsichtigt hatten.

Ein Befehl, wie der vorher beschriebene, weist den Computer an, alles was zwischen den Anführungszeichen steht, auf dem Bildschirm darzustellen.
Es gibt eine Längenbegrenzung für die Zeichen, die man zwischen die Anführungsstriche schreiben kann. Eine Meldung kann länger sein als eine Bildschirmzeile. Das heißt, daß sich eine Anweisung über mehr als eine Zeile erstrecken kann. Lange Befehle werden automatisch in der nächsten, darunterliegenden Bildschirmzeile weitergeführt. Tippen Sie folgendes Beispiel und drücken Sie RETURN:

```
PRINT "UNTER NORMALEN UMSTAENDEN,WAERE ER HEUTE GEKOMMEN"
```

Der Computer antwortet wie folgt:

```
UNTER NORMALEN UMSTAENDEN WAERE ER HEUTE GEKOMMEN
```

ATARI-BASIC akzeptiert bis zu drei Bildschirmzeilen in einem Befehl. Das sind in der Standardeinstellung 114 Zeichen. Bevor Sie diese Grenze erreichen, warnt Sie der Computer durch ein Klingelzeichen. Die vorgegebene Länge beinhaltet auch den PRINT-Befehl und die Interpunktion. Alles was Sie nach diesen 114 Zeichen tippen, wird vom Computer ignoriert, sobald sie die RETURN-Taste betätigen, um den Befehl zu beenden.

Berechnungen

In der direkten Betriebsart kann man den ATARI-Computer wie einen Rechner benutzen, denn er berechnet sofort die Lösung arithmetischer Operationen. Probieren Sie die folgenden Beispiele selbst aus:

```
READY                    Addition
 PRINT 4+6
10

READY                    Subtraktion
PRINT 500-437
63

READY                    Multiplikation
PRINT 100*23
2300

READY                    Division
PRINT 96/12
8

READY                    Bildung einer Potenz
PRINT 3^2
8.99999988

READY                    Kombination
PRINT 3*4*10-800
-680

READY
```

Der Computer gibt die richtige Lösung sofort in der darauffolgenden Zeile. Beachten Sie bitte, daß in diesen Beispielen keine Anführungsstriche verwendet wurden. Versuchen Sie es nun mit einer Rechenaufgabe, die Sie zwischen Anführungsstriche setzen.
Numerische Werte können eine maximale Anzahl von neun Ziffernstellen haben. Werte mit

mehr Stellen werden nach der neunten Ziffer abgeschnitten. Die Höchstzahl der ausgedruckten Stellen hängt von der Anzahl der Ziffern vor und nach dem Dezimalpunkt ab. Die folgenden Beispiele sollen Ihnen diesen Vorgang veranschaulichen:

```
PRINT 12.34567896
12.3456789

READY
PRINT 12.34567894
12.3456789

READY
PRINT 1234567895
1234567890

READY
```

Wenn Sie Ihre eigenen arithmetischen Operationen in der direkten Betriebsart eingeben, werden Sie feststellen, daß die Resultate in manchen Fällen in der Exponential-Schreibweise ausgegeben werden.

```
PRINT 123456789123
1.23456789E+11

READY
```

Sollten sie diese Schreibweise nicht verstehen, dann belassen Sie es im Moment bei einfachen Rechenaufgaben. Sie werden später in diesem Kapitel mehr über numerische Werte und die Exponential-Schreibweise erfahren.

Der abgekürzte PRINT-Befehl

In ATARI-BASIC hat man die Möglichkeit, den PRINT-Befehl durch ein Fragezeichen (?) abzukürzen. Probieren Sie die folgenden Beispiele selbst aus:

```
?"DIE ZEIT LAEUFT"
DIE ZEIT LAEUFT

READY
?13-46*6
-263

READY
```

Fehlermeldungen

Die Meldung, die der Computer ausgibt, wenn er entdeckt, daß er einer Situation nicht gewachsen ist, haben wir schon vorher in diesem Kapitel erwähnt. Es erscheint die Meldung ERROR- gefolgt von der fehlerhaften Anweisung. Es gibt jedoch verschiedene Formen von Fehlermeldungen. Wenn der ATARI-Computer glaubt, er wüßte um welche Art von Fehler es sich handelt, gibt er eine Diagnose in Form von einer Nummer aus. In diesem Beispiel soll die Zahl 1 durch 0 dividiert werden:

```
?1/0

ERROR- 11
```

Die exakte Übersetzung der Fehlerkennzahl 11 lautet "Zahlenbereich verlassen". Mit anderen Worten ergibt die Division durch 0 einen Wert, der zu groß für den Computer ist.
Die Kennzahl soll Ihnen die Fehlersuche erleichtern. Sie müssen zwar noch im Anhang A die Interpretation nachlesen, haben jedoch schon eine ungefähre Ahnung, was Sie falsch gemacht haben könnten. Leider sind die diagnostischen Fähigkeiten des Computers begrenzt. Eine Kennzahl wird für viele verschiedene Situationen benutzt. Deshalb können Sie auch keine exakte Analyse Ihres Fehlers erwarten. Der ATARI-Computer hat weniger als 60 Fehlerkennzahlen zur Verfügung, um eine Vielzahl von möglichen Fehlern und Fehlerkombinationen zu diagnostizieren.

Leerzeichen

Haben Sie sich schon gefragt, wo man in einer Zeile Leerzeichen setzt und wo nicht? In ATARI-BASIC sollten Sie auch mit den Zwischenräumen achtsam umgehen. Die sicherste Methode ist, den Stil zu kopieren, den wir in unseren Beispielen verwenden. Im allgemeinen sollten Sie Leerzeichen dort benutzen, wo es hilft, eine Zeile lesbarer zu machen. Verwenden Sie jeweils nur einen Zwischenraum, denn einige Beispiele haben gezeigt, daß zahlreiche Leerzeichen dem Computer in manchen Fällen Schwierigkeiten bereiten können. Es gibt jedoch eine Ausnahme, bei der Sie Zwischenräume benutzen können, wo und wieviele Sie möchten: zwischen den Anführungsstrichen des PRINT-Befehls. Wenn Sie eine Programmzeile tippen und nicht sicher sind, ob Sie ein Leerzeichen einfügen sollen oder nicht, schreiben Sie einfach weiter und beenden die Zeile wie gewohnt. Das Schlimmste, was passieren könnte, ist, daß der Computer eine Fehlermeldung ausgibt und Sie die Zeile korrigieren müssen.

Anweisungen, Textzeilen und Programme

Ein Programm besteht aus einem oder mehreren Befehlen, die den Computer mit der exakten und vollständigen Beschreibung der Aufgabe, die er ausführen soll, versorgen. Ist diese Aufgabe kurz und einfach, dann kann auch das Programm kurz und einfach sein. Die Direkteingaben, mit denen wir uns bis jetzt befaßt haben, sind alles kleine, einfache Programme. Jedes beinhaltet nur eine Anweisung – einen Befehl für den Computer. Die meisten Programme umfassen aber 10, 100, 1000 oder sogar mehr Befehle. Beachten Sie die folgenden Anweisungen:

```
PRINT "KEIN HAUS OHNE MAUS"
KEIN HAUS OHNE MAUS

READY
PRINT "KEIN SCHEUER OHNE KORN"
KEIN SCHEUER OHNE KORN


READY
PRINT "KEIN ROS OHNE DORN"
KEIN ROS OHNE DORN

READY
```

Jede dieser Direkteingaben schreibt eine Textzeile auf den Bildschirm. Alle drei Programme enthalten jeweils einen Befehl pro Zeile.

ATARI-BASIC akzeptiert es, wenn Sie mehr als nur eine Anweisung in die Zeile schreiben. Um zwei Befehle voneinander zu trennen, verwendet man einen Doppelpunkt. Vergleichen Sie nun die folgenden Programme mit den oben vorgestellten Beispielen:

```
PRINT "KEIN HAUS OHNE MAUS":PRINT "KEIN SCHEUER OHNE KORN":PRINT
"KEIN ROS OHNE DORN"
KEIN HAUS OHNE MAUS
KEIN SCHEUER OHNE KORN
KEIN ROS OHNE DORN

READY
```

Dieses einzeilige Programm, das einen Dreier-Befehl beinhaltet, druckt die gleichen drei Textzeilen aus, wie die vorhergehenden drei Ein-Befehl-Programme.

Programm-, logische und dargestellte Zeilen

Es gibt keine Grenze für die Anzahl von Befehlen, die man in einer Programmzeile unterbringen kann. Denken Sie jedoch daran, daß in der Standardeinstellung eine Zeile nicht länger als 114 Zeichen sein kann. Wenn Sie das 107. Zeichen einer langen Programmzeile tippen, erinnert Sie der Computer durch ein Klingelzeichen daran, daß Sie die Grenze erreichen. Alles, was Sie nach dem 114. Zeichen schreiben, wird vom Computer ignoriert, sobald sie RETURN drücken. Sie sehen also, daß die Möglichkeiten eines einzeiligen Programms begrenzt sind.
Der ATARI-Computer behandelt jede Programmzeile als eine zusammengehörige Zeile, selbst wenn sie mehr als eine Bildschirmzeile belegt. Eine Programmzeile ist ein Beispiel für eine logische Zeile. Die kürzeste logische Zeile beinhaltet nur ein Zeichen, während die längste logische Zeile normalerweise 114 Zeichen haben kann (in Kapitel 4 lernen Sie, wie man diese Längenbegrenzung auf 120 Zeichen erweitern kann). Folglich beinhaltet jede logische Zeile eine, zwei oder auch drei dargestellte Zeilen (physikalische Zeilen). Die RETURN-Taste markiert das Ende einer logischen Zeile.

Ein einzeiliges Programm

Sie können den Computer mit einem einzeiligen Programm veranlassen, eine Menge von Aufgaben auszuführen. Sehen Sie sich das folgende Beispiel an:

```
FOR I=1 TO 722:?"A";:NEXT I:?"PUH!"
```

Im Moment sollten Sie sich keine Gedanken darüber machen, was diese neuen Befehle bewirken. Schreiben Sie die Zeile genau ab und drücken Sie RETURN. Wenn Sie alles korrekt eingetippt haben, füllen sich 19 Zeilen des Bildschirms mit dem Buchstaben A. In der 20. Zeile steht das Wort PUH!

```
FOR I=1 TO 722:?"A";:NEXT I:?"PUH!"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
PUH!

READY
```

Die Programmzeile ist noch immer oben auf dem Bildschirm dargestellt, denn das Programm weist den Computer an, nur 19 Reihen des 24-zeiligen Schirms mit dem Buchstaben A zu füllen. Dadurch wird die Programmzeile ganz nach oben gerollt, jedoch nicht vom Bildschirm geschoben.
Nachdem das Programm ausgeführt ist, erscheinen wieder die Meldung READY und der Cursor.

Programmier-Betriebsart

Die Art des Programmierens, die wir bis jetzt ausprobiert haben, ist zwar lehrreich und interessant, aber die Möglichkeiten, die man mit der Direkteingabe hat, sind ziemlich begrenzt. Ein Nachteil ist zum Beispiel, daß man das Programm jedes Mal, wenn man es wieder benutzen möchte, neu eintippen muß. Sie werden zwar bald Editiertechniken kennenlernen, die es Ihnen ermöglichen, ein Programm, solange es auf dem Bildschirm steht, wiederzubenutzen. Aber auch hierbei gibt es Einschränkungen.
Was Sie also brauchen, ist eine Möglichkeit, bei der Sie mehrere Programmzeilen eingeben können, ohne daß jede Zeile sofort ausgeführt wird. Auf diese Weise können Sie Programme schreiben, die umfangreichere Aufgaben ausführen, als das ein einzeiliges Programm könnte.
Hierfür bietet sich die Programmier-Betriebsart an. Sie wird auch indirekte oder verzögerte Betriebsart genannt. In dieser Betriebsart prüft und speichert der Computer die Programmeingaben, führt die Anweisungen des Programms jedoch erst aus, wenn Sie ihm den Befehl dazu erteilen.

Programmausführung

Das Programm wird durchgeführt bzw. läuft, wenn der Computer die Anweisungen, die das Programm beinhaltet, ausführt. Bei der Direkteingabe wird jede Programmzeile ausgeführt, sobald Sie RETURN drücken. Anders ist es bei der Programmeingabe. Mit dem Befehl RUN weisen Sie den Computer an, das Programm durchzuführen, und jedes Mal wenn Sie den Befehl wiederholen, wird das Programm erneut ausgeführt.

Löschen eines Programms

Der ATARI-Computer speichert alle Eingaben, die in der Programmier-Betriebsart geschrieben sind. Bevor Sie ein neues Programm tippen, müssen Sie das alte mit dem Befehl NEW aus dem Speicher löschen. Ansonsten werden die Programmzeilen des alten und des neuen Programms miteinander vermischt.

Beenden eines Programms

Der Befehl END bedeutet in BASIC, die Durchführung des Programms zu stoppen. Deshalb sollte dieser Befehl auch die letzte Anweisung sein, die vom Programm vollzogen wird. ATARI-BASIC benötigt am Ende eines Programms keinen END-Befehl, denn der Computer beendet die Ausführung automatisch, wenn er keine Anweisung mehr vorfindet. Trotzdem schließen versierte Programmierer ihre Programme immer mit einem END-Befehl ab.

Zeilennummern

Eine Zeilennummer ist eine ein- bis fünfstellige Zahl, die an den Anfang der Programmzeile gesetzt wird. Diese Nummer ist der einzige sichtbare Unterschied zwischen einer Programmeingabe und einer Direkteingabe.
Versuchen Sie die folgenden Beispiele selbst:

```
NEW

READY
10 PRINT "HEUTE IST SAMSTAG"
20 END
RUN
HEUTE IST SAMSTAG

READY
```

Jede Zeilennummer muß unterschiedlich sein. Zwei Programmzeilen können nicht die selbe Nummer haben. Wenn Sie eine Zeilennummer noch einmal verwenden, erinnert sich der Computer nur an die Programmzeile, die Sie zuletzt eingegeben haben.
Sie werden das besser verstehen, wenn Sie die nachfolgenden Beispiele eingegeben haben:

```
NEW

READY
10 PRINT "ERSTE ZEILE 10"
10 PRINT "ZWEITE ZEILE 10"
20 END
RUN
ZWEITE ZEILE 10

READY
```

Die Zeilennummern bestimmen die Reihenfolge der Programmzeilen in einem BASIC-Programm. Die erste auszuführende Zeile muß die niedrigste Zeilennummer haben, während die letzte Zeile die höchste Nummer haben muß. Der ATARI-Computer ordnet die Zeilen anhand der Zeilennummern, selbst wenn Sie die Programmzeilen nicht in der richtigen Reihenfolge eingetippt haben. Beachten Sie das folgende Programm, bei dem die Zeilennummern durcheinander eingegeben wurden:

```
NEW

READY
30 PRINT "SONNTAGS"
10 PRINT "MONTAGS"
20 PRINT "UND"
```

```
40 PRINT "IST"
60 PRINT "SPRECHSTUNDE"
50 PRINT "KEINE"
70 END
RUN
MONTAGS
UND
SONNTAGS
IST
KEINE
SPRECHSTUNDE

READY
```

Um Ihnen zu beweisen, daß der ATARI-Computer Ihre Programmeingaben auch wirklich gespeichert hat, löschen Sie den Bildschirm mit der CLEAR-Taste (CTRL-<). Danach geben Sie wieder RUN ein.

```
RUN
MONTAGS
UND
SONNTAGS
IST
KEINE
SPRECHSTUNDE

READY
```

Es ist ganz einfach, einem Programm, das sich im Speicher befindet, Zeilen hinzuzufügen. Sie können die Zeilen an den Anfang, ans Ende oder irgendwo in die Mitte setzen. Achten Sie jedoch darauf, eine Zeilennummer zu wählen, die an der Stelle liegt, an der Sie die neue Zeile postieren möchten. Angenommen, Sie wollen eine zusätzliche Zeile an den Anfang des letzten Programmbeispiels setzen. Solange Sie nicht den Befehl NEW getippt haben, befindet sich das Programm noch im Speicher. Die niedrigste Zeilennummer war die 10. Also wird jede Programmzeile, die mit einer kleineren Zahl als 10 beginnt, an den Anfang gesetzt. Fügen Sie die folgende Zeile ein:

```
5 PRINT "MITTWOCHS,"
RUN
MITTWOCHS,
MONTAGS
UND
SONNTAGS
IST
KEINE
SPRECHSTUNDE

READY
```

Das ließ sich ganz einfach erledigen, weil das Originalprogramm mit der Nummer 10 und nicht mit 0 angefangen hat. Es ist immer besser, die Programmeingaben mit einer hohen Zeilennummer zu beginnen und auch zwischen den Zeilen genug Platz zu lassen. Dadurch können Sie später ohne Probleme Programmzeilen einfügen.

Programmzeilen mit mehreren Befehlen

Wie sie wissen, kann eine Programmzeile mehr als einen Befehl enthalten. Der erste Be-

fehl steht hinter der Zeilennummer, danach folgt der zweite. Um die Anweisungen voneinander zu trennen, benutzt man einen Doppelpunkt.

Ausdruck der Programmzeilen

Um nachzuprüfen, welche Programmzeilen der Computer gespeichert hat, schreiben Sie den Befehl LIST. Probieren Sie das gleich aus. Das Programm wird auf dem Bildschirm dargestellt, es sei denn, Sie haben in der Zwischenzeit NEW getippt oder den Computer ausgeschaltet:

```
LIST

5 PRINT "MITTWOCHS,"
10 PRINT "MONTAGS"
20 PRINT "UND"
30 PRINT "SONNTAGS"
40 PRINT "IST"
50 PRINT "KEINE"
60 PRINT "SPRECHSTUNDE"
70 END

READY
```

Was Sie jetzt sehen, ist der Programmausdruck. Es gibt unterschiedliche LIST-Befehle, die Ihnen ermöglichen, mehrere oder auch nur eine Zeile ausdrucken zu lassen. Diese Wahlfunktion ist besonders praktisch, wenn Sie ein langes Programm listen wollen, das nicht im Ganzen auf den Bildschirm paßt.
Nun schreiben Sie den Befehl LIST10. Sie sehen, daß die Zeile 10 aus dem letzten Beispielprogramm dargestellt wird:

```
LIST 10

10 PRINT "MONTAGS"

READY
```

Wenn Sie mehrere aufeinanderfolgende Programmzeilen ausdrucken möchten, müssen Sie die Anfangs- und Endnummer spezifizieren, wie Sie aus unserem nächsten Beispiel ersehen können:

```
LIST 20,40

20 PRINT "UND"
30 PRINT "SONNTAGS"
40 PRINT "IST"

READY
```

ATARI-BASIC druckt alle Programmzeilen einschließlich der vorher bestimmten Zeile aus. Desgleichen haben Sie die Möglichkeit, den Computer anzuweisen, alle Programmzeilen ab einer spezifischen Zeilennummer bis zum Programmende ausdrucken zu lassen. Nachfolgend sehen Sie Beispiele für zwei Möglichkeiten des LIST-Befehls:

```
LIST 0,10

5 PRINT "MITTWOCHS,"
10 PRINT "MONTAGS"

READY
LIST 30,32767

30 PRINT "SONNTAGS"
40 PRINT "IST"
50 PRINT "KEINE"
60 PRINT "SPRECHSTUNDE"
70 END

READY
```

Unterbrechen des Programmausdrucks

Man kann den Ausdruck eines Programms unterbrechen, indem man die Taste BREAK drückt. Das ist besonders praktisch, um eine endlose Ausgabe eines langen Programms zu stoppen.
Um den Ausdruck eines Programms vorübergehend zu blockieren, drückt man die Tasten CTRL-1. Der Ausdruck wird fortgeführt, wenn man nochmals CTRL-1 tippt. Man kann also mit CTRL-1 die Geschwindigkeit des Ausdrucks selbst steuern.

Laden und Speichern eines Programms

Der ATARI 410 Programm-Rekorder ermöglicht es Ihnen, Programme außerhalb des Computers sicherzustellen und sie später wieder in den Speicher einzulesen.
Angenommen, im Speicher befände sich folgendes Programm:

```
10 ?"DAS ALTE LIED UND LEID,"
20 ?"DASS DIE ERKENNTNIS ERST GEDEIHT,"
30 ?"WENN MUT UND KRAFT VERRAUCHEN;"
40 ?"DIE JUGEND KANN, DAS ALTER WEISS;"
50 ?"DU KAUFST NUR UM DES LEBENS PREIS"
60 ?"DIE KUNST, DAS LEBEN RECHT ZU BRAUCHEN."
70 END
```

Legen Sie eine Kassette in den Programm-Rekorder ein und spulen Sie das Band bis zum Anfang zurück. Dann tippen Sie folgenden Befehl:

```
CSAVE
```

Zwei Klingelsignale machen Sie darauf aufmerksam, sowohl die RECORD- als auch die PLAY-Taste auf dem Rekorder zu drücken. Nachdem Sie die RETURN-Taste gedrückt haben, beginnt die Kassette aufzuzeichnen. Während der ersten 20 Sekunden hören Sie ein schnarrendes Klingelgeräusch. Danach beginnt der eigentliche Speichervorgang. Dieser wird durch Klingelgeräusche angezeigt. Wenn sich der Ton verändert und anschließend

verstummt, ist der Speichervorgang beendet.
Jetzt schreiben Sie den Befehl NEW, um das Programm aus dem Computerspeicher zu löschen. Mit dem Befehl LIST können Sie sich vergewissern, daß sich das Programm nicht mehr im Speicher befindet.
Um das Programm von der Kassette wieder in den Computer zu laden, spulen Sie das Band bis Anfang zurück. Dann tippen Sie folgenden Befehl:

```
CLOAD
```

Der Computer fordert Sie mit einem Klingelsignal auf, die PLAY-Taste auf dem Programm-Rekorder und danach die RETURN-Taste auf der Tastatur zu drücken. Nach einigen Sekunden hören sie ein paar knackende Geräusche aus dem Fernsehlautsprecher. Diese Töne zeigen an, daß das Programm in den Computerspeicher geladen wird. Wenn die Geräusche verstummen, ist der Ladevorgang beendet und das Band stoppt. Nun schreiben Sie den Befehl LIST. An dem Programmausdruck sehen Sie, daß der Ladevorgang geklappt hat.
In Kapitel 5 werden andere Möglichkeiten beschrieben, Programme auf der Kassette sicherzustellen. In Kapitel 7 wird erklärt, wie man Programme auf der Diskette speichert.

Speichern mehrerer Programme auf einer Kassette

Sie haben vielleicht bemerkt, daß Sie lediglich eine geringe Bandlänge benötigten, um das Beispielprogramm auf der Kassette zu speichern. Ein längeres Programm nimmt natürlich mehr Platz in Anspruch, aber normalerweise reicht eine Kassette für mehrere BASIC-Programme. Sie können Ihre Programme der Reihenfolge nach auf dem Band speichern.
Das Laden des zweiten, dritten und der folgenden Programme ist nicht so einfach wie das Laden des ersten. Zuerst spulen Sie das Band bis zum Anfang zurück. Dann führen Sie den Ladevorgang mit dem ersten Programm durch, dann mit dem zweiten und anschliessend erst können Sie das dritte Programm laden. Das ist allerdings ziemlich umständlich.
Sie können diesen Prozess erheblich beschleunigen, indem Sie den Bandzähler des Programm-Rekorders benutzen. Bevor Sie ein Programm speichern, spulen Sie das Band bis zum Anfang zurück und stellen den Zähler auf 0. Nachdem Sie das erste Programm gespeichert haben, notieren Sie die Zählernummer, die zugleich den Anfang des zweiten Programms markiert. Nach dem Speichern des zweiten Programms notieren Sie sich wieder die Zählernummer (für den Anfang des dritten Programms).
Wenn Sie das zweite Programm laden möchten, spulen Sie zuerst das Band bis zum Anfang zurück und stellen den Bandzähler auf 0. Jetzt drücken Sie die ADVANCE-Taste und spulen das Band soweit vor, bis das zweite Programm bereit steht. Mit Hilfe des Bandzählers und der notierten Zählernummer können Sie die gewünschte Stelle schnell wiederfinden. Haben Sie das Band zu weit vorwärts gespult, so können Sie mit der REWIND-Taste zur Startnummer des zweiten Programms zurückspulen. Nun tippen sie den Befehl CLOAD, und das Programm wird in den Computer geladen.

Fortgeschrittene Editier-Möglichkeiten

In Kapitel 2 wurde erklärt, wie man Tippfehler schnell korrigieren kann. Nachfolgend sehen Sie eine Liste mit einfachen Editier-Möglichkeiten:

- Die BACK S-Taste bewegt den Cursor zeichenweise rückwärts, wobei das Getippte gelöscht wird.
- Die ←Taste bewegt den Cursor eine Stelle nach links, ohne das Programm oder die Anzeige zu ändern.

- Die →Taste bewegt den Cursor eine Stelle nach rechts, ohne das Programm oder die Anzeige zu ändern.
- Die TAB-Taste bewegt den Cursor zu der nächsten Tabulatorstelle, ohne das Programm oder die Anzeige zu ändern.
- Die BREAK-Taste unterbricht die Zeile und setzt den Cursor an den Anfang der nächsten Zeile.
- Die CLEAR-Taste löscht den Bildschirm und setzt den Cursor in die linke, obere Schirmecke.

Diese Editier-Möglichkeiten sind sowie bei der Direkteingabe als auch bei der Programmeingabe sehr hilfreich. Anschließend werden Sie noch andere Editier-Möglichkeiten kennenlernen, die Ihnen speziell in Programmeingaben Korrekturen erleichtern.

Löschen von Programmzeilen

Um eine BASIC-Zeile zu löschen, tippt man nur die entsprechende Zeilennummer und drückt RETURN. Am Programmausdruck können Sie erkennen, daß sich die Zeile nicht mehr im Programm befindet. Hier ist ein Beispiel:

```
NEW

READY
100 PRINT "ES IST NICHT ALLES GEIST, WAS GLAENZT"
110 PRINT "IDEALE SIND EIN LEUCHTTURM, KEIN HAFEN"
120 PRINT "HOEHEPUNKTE LEBEN VON ANLAEUFEN"
130 PRINT "WER ANDERN EINE GRUBE GRAEBT, FAELLT SELBST HINEIN"
140 PRINT "EIN VORURTEIL IST EIN IRRTUM, DER WURZELN GESCHLAGEN
HAT"
150 END
110
130
LIST

100 PRINT "ES IST NICHT ALLES GEIST, WAS GLAENZT"
120 PRINT "HOEHEPUNKTE LEBEN VON ANLAEUFEN"
140 PRINT "EIN VORURTEIL IST EIN IRRTUM, DER WURZELN GESCHLAGEN
HAT"
150 END

READY
```

Hinzufügen von Programmzeilen

Man kann jederzeit, egal in welcher Reihenfolge, neue Programmzeilen eingeben. Die Zeilennummern bestimmen ihre Position im Programm. Der ATARI-Computer fügt sie automatisch in das bereits gespeicherte Programm ein. Versuchen Sie nun, die Zeile 110 in das oben gezeigte Beispiel wiedereinzufügen.

```
110 PRINT "IDEALE SIND EIN LEUCHTTURM, KEIN HAFEN"
LIST

100 PRINT "ES IST NICHT ALLES GEIST, WAS GLAENZT"
110 PRINT "IDEALE SIND EIN LEUCHTTURM, KEIN HAFEN"
120 PRINT "HOEHEPUNKTE LEBEN VON ANLAEUFEN"
140 PRINT "EIN VORURTEIL IST EIN IRRTUM, DER WURZELN GESCHLAGEN
HAT"
150 END
```

Ändern der Programmzeilen

Der einfachste Weg, eine Programmzeile zu ändern, ist, sie noch einmal zu schreiben. Aber das hat auch Nachteile, denn, wie Sie wissen, ist das Tippen der Zeilen eine zeitraubende Angelegenheit. Außerdem ist die Wahrscheinlichkeit, Tippfehler zu machen, ziemlich groß. Glücklicherweise gibt es schnellere Wege, Programmzeilen, die der Computer bereits gespeichert hat, zu ändern. Das ist möglich, da alles, was auf dem Bildschirm steht, "live" ist. Sie können editieren, indem Sie die Taste CTRL gleichzeitig mit einer der Steuerungstasten drücken und so den Cursor in allen Richtungen bewegen können. Das ermöglicht Ihnen, den Cursor auf jeden beliebigen Punkt des Bildschirms zu fahren. Jetzt kann man Zeilen nach Belieben einsetzen, löschen oder durch andere ersetzen.

Zeilenausdruck zum Editieren

Um etwas zu editieren, egal ob es sich um eine Direkteingabe, eine Programmeingabe oder auch um die Antwort auf eine Frage des Computers handelt, muß die Zeile auf dem Bildschirm stehen. Wenn Sie eine Direkteingabe ändern möchten und diese nicht mehr auf dem Bildschirm dargestellt ist, haben Sie Pech. Sie müssen die Zeile noch einmal schreiben. Dagegen können Sie Programmeingaben mit dem Befehl LIST ausdrucken lassen. Geben Sie dabei die Anfangs- und Endzeilennummern an, damit die Zeile, die Sie ändern möchten, auch wirklich auf dem Bildschirm steht. Sobald die Programmzeile dann dargestellt ist, können Sie diese umgestalten.

Bewegen des Cursors

Es gibt sieben Tasten, um den Cursor zu bewegen. Die BACK S, →, ← und die TAB-Tasten wurden schon erklärt, die Leertaste ist eine weitere. Sie bewegt den Cursor wie die →Taste, ersetzt jedoch jedes Zeichen, über das der Cursor fährt, durch ein Leerzeichen. Die ↑ und ↓Tasten wurden in Kapitel 2 erwähnt. Die ↑Taste führt den Cursor eine Zeile nach oben. War der Cursor bereits in der obersten Zeile, dann bleibt er in derselben Spalte und springt in die unterste Zeile. Dagegen führt die ↓Taste den Cursor eine Zeile nach unten. War der Cursor bereits in der untersten Zeile, dann bleibt er in derselben Spalte und springt in die oberste Zeile.

Quittieren der Programmänderungen

Nachdem Sie eine Programmzeile modifiziert haben, müssen Sie RETURN drücken, um dem Computer diese Veränderung mitzuteilen. Die Zeile bleibt im Programm unverändert, wenn Sie mit dem Cursor lediglich in eine andere Zeile fahren, ohne die RETURN-Taste zu betätigen. Dadurch wird nur der Bildschirmtext geändert, ohne daß der Computer die Änderungen berücksichtigt. Dabei ist es gleich, an welcher Stelle der modifizierten Zeile der Cursor steht, wenn Sie RETURN drücken.

Annulieren einer Programmänderung

Es gibt drei verschiedene Wege, eine Änderung, die Sie schon auf dem Bildschirm vorgenommen haben, zu widerrufen. Die folgenden Möglichkeiten haben Sie aber nur, wenn Sie RETURN noch nicht gedrückt haben:

- Die BREAK-Taste drücken.
- Die Steuerungstasten benutzen, um den Cursor aus der Programmzeile zu bewegen.
- Die CLEAR-Taste (CTRL-<) drücken, um den Bildschirm zu löschen.

Zeichen ersetzen

Es ist ganz einfach, ein Zeichen durch ein anderes zu ersetzen. Sie fahren mit dem Cursor über das entsprechende Zeichen und schreiben die neue Eingabe einfach darüber. Zum Beispiel:

```
READY
 100 ?"GESCHAETZTE ABFLUGSZEIT"
```

Sie postieren den Cursor über dem A und schreiben ANKUNFTSZEIT:

```
100 ?"GESCHAETZTE ANKUNFTSZEIT"
```

Vergessen Sie nicht, RETURN zu drücken, um die Änderung zu bewirken.

Zeichen löschen

Sie haben drei Möglichkeiten, Zeichen zu löschen. Bei der ersten fahren Sie mit dem Cursor auf das Zeichen, das Sie entfernen möchten, und drücken die DELETE-Taste (CTRL-BACK S). Das Zeichen wird gelöscht und der Rest der Programmzeile rückt um eine Stelle nach links. Zum Beispiel:

```
10 PRINT "WENN ES KLOPFT KOMMEN SIIEE HEREIN"
```

Drücken Sie die DELETE-Taste (CTRL-BACK S) zweimal und sehen Sie was passiert:

```
10 PRINT "WENN ES KLOPFT KOMMEN SIE HEREIN"
```

Die beiden anderen Möglichkeiten, Zeichen zu löschen, haben Sie durch die BACK S- und die Leertaste. Die BACK S-Taste bewegt den Cursor zeichenweise rückwärts, während die Leertaste den Cursor vorwärts bewegt. Dabei wird das Getippte gelöscht und durch Leerzeichen ersetzt.

Zeichen einfügen

Wenn Sie neue Zeichen einfügen möchten, müssen Sie zuerst den erforderlichen Freiraum durch Einsetzen von Leerzeichen schaffen. Das erreichen Sie mit den INSERT-Tasten (CTRL->). Anschließend können Sie die Zeichen hier einfügen, ohne den Text zu verändern. Jedes Leerzeichen, das Sie einfügen, schiebt den Rest der Programmzeile um eine Stelle nach rechts. Wenn dadurch das letzte Zeichen der Zeile über den Bildschirmrand hinausgeschoben wird, so produziert der Computer automatisch eine neue Bildschirmzeile, in der das Zeichen dann dargestellt wird. Sehen Sie dazu die folgenden Beispiele:

```
10 PRINT "PREIS PRO KILO"
20 PRINT "GESAMTGEWICHT"
30 PRINT "SUMME"
```

Bevor Sie der Zeile 10 weiteren Text hinzufügen, drücken Sie die INSERT-Tasten (CTRL->) 20 mal.

```
10 PRINT "PREIS PRO KILO
        "
20 PRINT "GESAMTGEWICHT"
30 PRINT "SUMME"
```

Sie konnten beobachten, wie der Computer zwischen den Zeilen 10 und 20 eine neue Bildschirmzeile eingefügt hat. Tippen Sie nun:

```
10 PRINT "PREIS PRO KILO, IN SUEDDEUTSCHLAND"
20 PRINT "GESAMTGEWICHT"
30 PRINT "SUMME"
```

Vergessen Sie nicht RETURN zu drücken, um die modifizierte Zeile zu speichern.

Automatische Zeichenwiederholung

Jede Taste wiederholt automatisch ihre Funktion, wenn sie länger als eine Sekunde gehalten wird. Damit können Sie das Editieren wesentlich beschleunigen.

Nochmalige Ausführung der Direkteingaben

Die Tatsache, daß alles auf dem Bildschirm Dargestellte "live" ist, ermöglicht es Ihnen, jede Direkteingabe wiederholt ausführen zu lassen, wobei Sie die Wahl haben, ob Sie diese vorher verändern möchten oder nicht. Voraussetzung dafür ist natürlich, daß die Eingabe noch auf dem Bildschirm steht.
Wenn Sie etwas editieren wollen, setzen Sie den Cursor auf die entsprechende Stelle, und mit Hilfe der Steuerungstasten (CTRL--, CTRL-=, CTRL-+ und CTRL-*) können Sie Zeichen löschen, ersetzen oder einfügen. Danach, der Cursor muß noch auf der Zeile stehen, drücken Sie RETURN und die Anweisungen werden ausgeführt.
Das folgende Beispiel soll Ihnen diese Funktionen veranschaulichen. Die untenstehende Direkteingabe rechnet das Volumen eines 10x5x4 Meter großen Raumes aus:

```
PRINT "DAS VOLUMEN BETRAEGT=";10*5*4
DAS VOLUMEN BETRAEGT=200

READY
```

Sie können diese Direkteingabe ohne Probleme modifizieren, um zum Beispiel die Werte 10x5x8 einzusetzen. Sie postieren zuerst den Cursor an den Anfang der Zeile und drücken die →Tasten (CTRL-*). Durch das Halten der Tasten wird der Cursor veranlaßt, auf der Zeile entlangzulaufen. Wenn er die Ziffer 4 erreicht hat, lassen Sie beide Tasten auf einmal los. Sollte der Cursor links oder rechts neben der betreffenden Ziffer stehen, dann benutzen Sie die Cursorsteuerungstasten, um ihn auf die gewünschte Position zu setzen. Anstatt dessen haben Sie aber auch die Möglichkeit, vom Beginn der Zeile an 36 mal die →Taste zu betätigen, um den Cursor auf die gewünschte Ziffer zu setzen.
Noch eine andere Möglichkeit wäre, die TAB-Taste 4mal zu drücken und die ←Taste 7mal. Im übrigen bleibt es Ihrer Wahl überlassen, welche Taste Sie zum Bewegen des Cursors vorziehen.
Ersetzen Sie nun die Zahl 4 durch ein 8 und drücken Sie RETURN:

```
PRINT "DAS VOLUMEN BETRAEGT=";10*5*8
DAS VOLUMEN BETRAEGT=400

READY
```

Programmiersprachen

Programmiersprachen sind entwickelt worden, um die Kommunikation zwischen Mensch und Computer zu ermöglichen. Es gibt viele verschiedene Programmiersprachen. BASIC zum Beispiel ist eine Allzweck-Programmiersprache. Dagegen wurden andere problemorientierte Sprachen entwickelt, die es ermöglichen, Programme für spezielle Anwendungen zu schreiben. Es gibt Programmiersprachen für den kaufmännischen und den wissenschaftlichen Bereich, zur Herstellung von speziellen Graphiken, zur Textverarbeitung und so weiter. Programmiersprachen gibt es, ebenso wie Fremdsprachen, in vielen Variationen. Neben BASIC sind ebenfalls gebräuchliche Programmiersprachen zum Beispiel: FORTRAN, Pascal, COBOL, PL-1 und FORTH.
ATARI-Computer sind so ausgelegt, daß sie mehrere Programmiersprachen verstehen; unter anderem BASIC und FORTH. Dieses Buch beschäftigt sich mit dem Programmieren in ATARI-BASIC.
Egal um welche Sprache es sich handelt, jede Programmanweisung muß unter Berücksichtigung genau definierter Regeln formuliert werden. Diese Regeln werden als Syntax bezeichnet, wobei zu bemerken ist, daß jede Programmiersprache ihre eigene Syntax hat.
Ebenso wie gesprochene Sprachen haben auch Programmiersprachen Dialekte. Diese Dialekte unterscheiden sich untereinander geringfügig in ihrer Syntax. Der ATARI-Computer kennt verschiedene BASIC-Dialekte. ATARI bietet Microsoft-BASIC und Standard-ATARI-BASIC (das mit dem ATARI400/800-Computer geliefert wird) an. BASIC A+ ist im Fachhandel erhältlich.
Wenn das Programm in einem bestimmten BASIC-Dialekt formuliert wurde und der Computer die Anweisungen in einem anderen Dialekt erwartet, kann es passieren, daß das Programm nicht ordnungsgemäß läuft. Außerdem können BASIC-Programme, die für den ATARI-Computer geschrieben worden sind, in den meisten Fällen von anderen Computern nicht korrekt durchgeführt werden, selbst wenn diese in BASIC programmierbar sind. Wie auch immer, wenn Sie gelernt haben, Ihren ATARI-Computer in einem der BASIC-Dialekte zu programmieren, werden Sie keine Schwierigkeiten haben, auch die anderen Dialekte zu beherrschen.
Einige Syntaxregeln der Programmiersprache sind ganz einfach. Die Syntax der Additions- und Subtraktionsbeispiele am Anfang dieses Kapitels ist jedermann vertraut. Man muß kein Programmierer sein, um sie zu verstehen. Die meisten der Syntaxregeln jedoch werden Ihnen am Anfang ziemlich unsinnig erscheinen. Deshalb sollten Sie im Moment auch nicht nach einer vernünftigen Erklärung dafür suchen, denn die ergibt sich von allein, sobald Sie die Zusammenhänge verstehen. Warum wird zum Beispiel ein Sternchen (*) für

die Multiplikation benutzt? Normalerweise würden Sie dafür ein x schreiben. Der Computer jedoch kann zwischen dem Buchstaben "x" und dem "x", das eigentlich eine Multiplikation veranlassen sollte, keinen Unterschied feststellen. Deshalb hat man sich entschieden, in beinahe allen Programmiersprachen das * für die Multiplikation zu verwenden. Die Division wird durch das /-Zeichen repräsentiert. Es gibt keinen Grund für diese Wahl, aber es mußte ein Zeichen gefunden werden, das auf jedem Standard-Tastenfeld vorhanden ist.

BASIC-Elemente

Die meisten Syntaxregeln beziehen sich auf die individuellen Befehle. Die Syntax der BASIC-Anweisungen behandelt deren drei Hauptelemente im einzelnen: Zeilennummern, Daten und Befehle für den Computer. Es gibt auch ein paar Regeln, die sich auf das Programm im Ganzen beziehen, wie zum Beispiel die Reihenfolge der Anweisungen. Alle diese Regeln werden Sie nach und nach in diesem Kapitel lernen.

Mehr über Zeilennummern

Inzwischen wissen Sie schon einiges über Zeilennummern. Nach einem kurzen Rückblick werden wir mehr ins Detail gehen. Jede Zeilennummer darf nur einmal benutzt werden. Die Zeilennummern bestimmen die Reihenfolge, in der die Programmanweisungen ausgeführt werden. Die Zeile mit der niedrigsten Nummer wird zuerst bearbeitet, die Zeile mit der höchsten zum Schluß.
Standard-ATARI-BASIC akzeptiert ein- bis fünfstellige ganzzahlige Zeilennummern zwischen 0 und 32767.

Zeilennummern als Adressen

Im wesentlichen dienen Zeilennummern dazu, Programmzeilen in einem Programm zu adressieren. Dieses Konzept ist sehr wichtig, da jedes Programm aus zwei Kategorien von Anweisungen besteht:

- Anweisungen, die Daten erzeugen oder modifizieren
- Anweisungen, die die Reihenfolge kontrollieren, in der die Befehle ausgeführt werden.

Es ist logisch, daß die Aufgaben, die das Programm beinhaltet, in einer ganz bestimmten Reihenfolge ausgeführt werden müssen. Welchen Sinn würde es denn geben, wenn der Computer die Anweisungen in einer Zufallsreihenfolge bearbeiten würde? Normalerweise beginnt die Programmdurchführung mit dem ersten Befehl, danach mit dem zweiten, gefolgt von dem dritten und so weiter (siehe Abbildung 3-1). Die meisten Programme werden jedoch nicht nach diesem Schema ausgeführt. Und nun werden die Zeilennummern wichtig. Sie können den Computer zum Beispiel anweisen, anstatt zur nächsten Zeile zu einer anderen Zeilennummer zu gehen und dort mit der Programmdurchführung fortzufahren (Abbildung 3-2).

Daten

Die Hauptaufgabe eines Computerprogramms besteht darin, eingegebene Daten zu bearbeiten und anschließend auszugeben. Deshalb ist es wichtig zu wissen, wie eine Programmiersprache die Daten handhabt. Anschließend werden Sie die verschiedenen Datentypen, die in einem ATARI-BASIC-Programm vorkommen, näher kennenlernen.

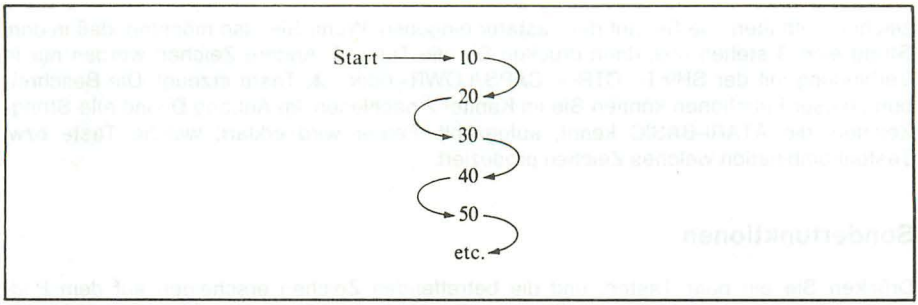


Abbildung 3-1: Das Programm wird gemäß der Reihenfolge der Zeilennummern ausgeführt

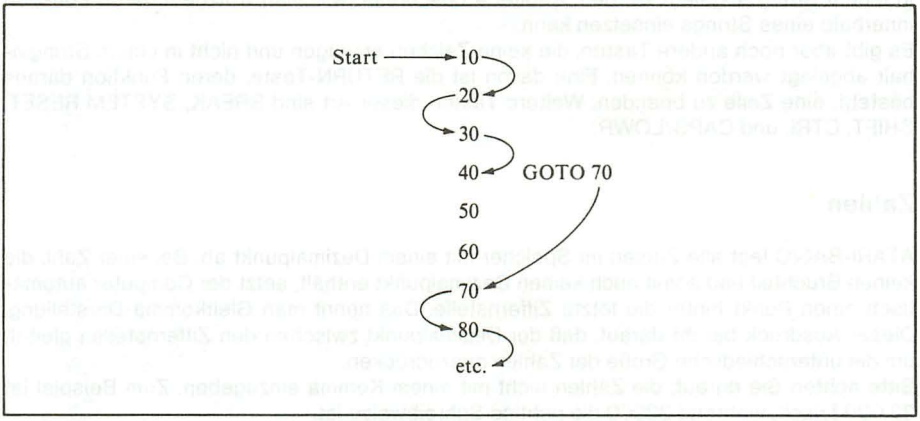


Abbildung 3-2: Die Programmdurchführung wird unterbrochen und in einer vorher angegebenen Zeile fortgesetzt

String

Als String oder Zeichenkette bezeichnet man ein oder mehrere Zeichen, die zwischen Anführungsstrichen stehen. Sie haben bereits Strings kennengelernt, die Sie mit dem Befehl PRINT auf dem Bildschirm dargestellt haben. Hier sind noch ein paar weitere String-Beispiele:

```
"SONNTAGS NIE"
"ADDIEREN SIE 40+181+324+837"
"UWE SCHMIDT"
"KARIN & PETER SCHUETZ"
"9.MAERZ 1957"
```

Es besteht keine spezielle Längenbegrenzung für Strings. Allerdings kann sich in der direkten Betriebsart der String nicht über mehr als eine Programmzeile erstrecken. In Kapitel 4 wird die Möglichkeit aufgezeigt, wie man in der Programmier-Betriebsart Strings miteinander kombinieren kann. In dieser Betriebsart ist die maximale Stringlänge von der verfügbaren Speicherkapazität abhängig. Ein String, der keine Zeichen enthält, wird als Nullstring oder auch Leerstring bezeichnet. Ein String kann Buchstaben, Ziffern oder Sonder-

zeichen enthalten, die Sie auf der Tastatur eingeben. Wenn Sie also möchten, daß in dem String eine 3 stehen soll, dann drücken Sie die Taste 3. Andere Zeichen werden nur in Verbindung mit der SHIFT-, CTRL-, CAPS/LOWR- oder ⅄ Taste erzeugt. Die Beschreibung dieser Funktionen können Sie im Kapitel 2 nachlesen. Im Anhang D sind alle String-Zeichen, die ATARI-BASIC kennt, aufgezählt. Ferner wird erklärt, welche Taste bzw. Tastenkombination welches Zeichen produziert.

Sonderfunktionen

Drücken Sie ein paar Tasten, und die betreffenden Zeichen erscheinen auf dem Bildschirm. Normalerweise kann man alle diese Zeichen einem String zuordnen. Es gibt jedoch einige Ausnahmen, die sich auf die Sonderzeichen beziehen. Zum Beispiel die Pfeiltasten (CTRL--, CTRL-=, CTRL-+ und CTRL-*). Diese Cursorsteuerungstasten können nicht direkt in einem String abgelegt werden. Kapitel 4 beschreibt, wie man trotzdem deren Funktion innerhalb eines Strings einsetzen kann.
Es gibt aber noch andere Tasten, die keine Zeichen erzeugen und nicht in einem Stringinhalt abgelegt werden können. Eine davon ist die RETURN-Taste, deren Funktion daraus besteht, eine Zeile zu beenden. Weitere Tasten dieser Art sind BREAK, SYSTEM RESET, SHIFT, CTRL und CAPS/LOWR.

Zahlen

ATARI-BASIC legt alle Zahlen im Speicher mit einem Dezimalpunkt ab. Bei einer Zahl, die keinen Bruchteil und somit auch keinen Dezimalpunkt enthält, setzt der Computer automatisch einen Punkt hinter die letzte Ziffernstelle. Das nennt man Gleitkomma-Darstellung. Dieser Ausdruck beruht darauf, daß der Dezimalpunkt zwischen den Ziffernstellen gleitet, um die unterschiedliche Größe der Zahlen auszudrücken.
Bitte achten Sie darauf, die Zahlen nicht mit einem Komma einzugeben. Zum Beispiel ist 32,000 falsch, während 32000 die richtige Schreibweise ist.

Ganzzahlen

Ganzzahlen haben weder einen Bruchteil noch einen Dezimalpunkt. Sie können negativ (–) oder positiv (+) sein. Wenn einer Zahl kein Vorzeichen vorausgeht, behandelt sie der Computer als positiven Wert. ATARI-BASIC rechnet mit Ganzzahlen genauso wie mit Gleitkomma-Zahlen. Die folgenden Beispiele sind Ganzzahlen:

```
0
1
44
32699
- 15
```

Gleitkomma-Zahlen

Eine Gleitkommazahl kann eine Ganzzahl, eine Zahl und Dezimalbruch oder nur ein Dezimalbruch sein. Sie können negative oder positive Vorzeichen haben. Wird kein Vorzeichen angegeben, so werden sie vom Computer als positive Werte behandelt.
Nachfolgende sehen Sie einige Beispiele für Gleitkomma-Zahlen:

```
      5
   - 15
 65000
   161
     0
     0.5
     0.0165432
   - 0.0000009
     1.6
    24.0055
 - 64.2
```

Exponential-Schreibweise

ATARI-BASIC benutzt zur Darstellung sehr großer und sehr kleiner Zahlen die Exponential-Schreibweise. Jede Zahl, die mehr als zehn Ziffernstellen vor dem Dezimalpunkt hat, wird in der Exponential-Schreibweise ausgegeben. Außerdem wird jede Bruchzahl, die näher bei 0 liegt als ±0.01, in dieser Schreibweise dargestellt.
Eine Zahl, die in der Exponential-Schreibweise ausgegeben wird, hat folgendes Format:

±Zahl E±ee

Dieser Ausdruck bedeutet:

+ kann wahlweise positives oder negatives Vorzeichen sein.

Zahl ist eine Ganzzahl, ein Bruch oder eine Kombination daraus. Diese Ziffern bilden die Mantisse (auch Koeffizient genannt).

E Dieser Buchstabe ist die Abkürzung für Exponent.

± kann wahlweise positives oder negatives Vorzeichen sein.

ee ist der ein- bis zweistellige Exponent. Der Exponent bestimmt die Größe der Zahl dadurch, indem er angibt, um wievel Stellen der Dezimalpunkt verschoben werden muß. Beim positiven Vorzeichen wird der Punkt nach rechts gerückt, beim negativen nach links.

Nachfolgend sehen Sie ein paar Beispiele, die Werte in der üblichen und in der Exponential-Schreibweise darstellen.

übliche Schreibweise	**Exponential-Schreibweise**
1000000000	1E+09
0.000000001	1E-09
200	2E+02
- 12345678900	- 1.23456789E+10
- 0.00000123456789	- 1.23456789E-06

Wie Sie selber sehen, ist die Exponential-Schreibweise eine bequeme Art, sehr große und sehr kleine Zahlen auszudrücken.

Zahlenbereiche

Die kleinste mögliche Gleitkommazahl ist – 9.99999999+E97 und die größte mögliche ist 9.99999999+E97. Wenn eine Bruchzahl sich 0 um mehr als ±9.99999999–E98 nähert, wird sie in die Zahl 0 umgewandelt.

Auf- und Abrunden

In diesem Kapitel wurde schon erwähnt, daß bei Gleitkommazahlen nicht mehr als neun Ziffernstellen ausgegeben werden.
Wenn eine Zahl mehr als neun Stellen hat, so ersetzt der ATARI-Computer die restlichen Ziffern intern durch Nullen. Probieren Sie nun die Beispiele. Dabei werden Sie bemerken, daß große Zahlen in der Exponential-Schreibweise dargestellt werden.

```
PRINT 12345678989
1.23456789E+10

READY
?-123456789123456789
-1.23456789E+17

READY
?-150000475.75
-150000475

READY
?90000000.7558
90000000.7

READY
```

Bruchzahlen zwischen 1 und –1 unterliegen den gleichen Regeln. Bei Zahlen in diesem Bereich beginnt das Abzählen der neun Ziffernstellen erst nach dem Dezimalpunkt und zwar bei der ersten Ziffer, die nicht Null lautet:

```
PRINT .1234567899
0.123456789

READY
?-.123456789123456789
-0.123456789

READY
?-123456789.123456789
-123456789

READY
?.0000000090000000007558
9.00000007E-10

READY
```

Variablen

In den Beispielen, die wir bis jetzt besprochen haben, wurden ausschließlich konstante Werte benutzt. Meistens verwendet man jedoch ganze Datengruppen, die einer Variablen zugewiesen werden.
Wenn Ihnen bereits die Grundzüge der Algebra vertraut sind, werden Sie keine Schwierigkeiten haben, das Konzept von Variablen und Variablen-Namen zu verstehen. Hatten Sie jedoch noch nie etwas mit Algebra zu tun, dann stellen Sie sich den Variablen-Namen als Namen auf einem Briefkasten vor (Abb. 3-3). Man gibt den Namen des Briefkastens an, dem einige Daten zugewiesen werden sollen. Dadurch wird dann ein Wert in einer Variablen abgelegt bzw. gespeichert.
Alle Werte, die in einer Variablen abgelegt wurden, können wieder verändert werden, und hier liegt auch der große Vorteil der Variablen – sie können jeden beliebigen legalen Wert darstellen. Sie haben somit die Möglichkeit, den Wert einer Variablen während des Programmablaufs zu verändern. BASIC kennt verschiedene Befehle, mit denen Sie das durchführen können. Diese werden Sie nacheinander kennenlernen.

Variablen-Namen

Der Name, den man einer Variablen zuweist, darf so lang sein wie eine Programmzeile. Das erste Zeichen muß in jedem Fall ein Großbuchstabe sein, während die übrigen Zeichen aus Ziffern und Großbuchstaben bestehen können. Eine String-Variable wird durch ein Dollarzeichen ($) gekennzeichnet, das an der letzten Stelle ihres Namens steht. In Abbildung 3-4 wird der Aufbau der Variablen-Namen illustriert.

String-Variablen

Bevor Sie einer String-Variablen Zeichen zuordnen können, müssen Sie dem Computer die maximale Länge des Strings mitteilen. Das führen Sie mit Hilfe des DIM-Befehls durch, den wir aber erst später erklären werden. Sollten Sie die Dimensionierung versäumen, so gibt der Computer eine Fehlermeldung aus, sobald Sie in der Variablen Daten speichern wollen.
Man kann einer String-Variablen beliebig viele Daten zuweisen. Die einzige Beschränkung besteht darin, daß die verfügbare Speicherkapazität nicht überschritten werden kann. Leerzeichen innerhalb eines Strings zählen genauso wie jedes andere Zeichen und müssen somit bei der Dimensionierung berücksichtigt werden. Die Leerzeichen, die am Ende des Strings stehen, auch nachfolgende Leerzeichen genannt, zählen ebenso.
Hier sehen Sie legale und illegale Variablennamen:

Legal	**Illegal**
A$	$
VORNAME$	9$
TEIL1$	FOLG.NAMEN$
ANTWORTEN$	a$
X8$	Name$

Numerische Variablen

Numerische Variablen können ganzzahlige Werte oder Gleitkomma-Zahlen beinhalten. Die Größenordnung ist jedoch auf –10 bis +10 begrenzt. Wollen Sie Werte außerhalb dieser Bandbreite ablegen, erhalten Sie eine Fehlermeldung. ATARI-BASIC wandelt Zahlen, die sich um mehr als $\pm 9.99999999 \cdot 10^{-98}$ der Null nähern, in 0 um.

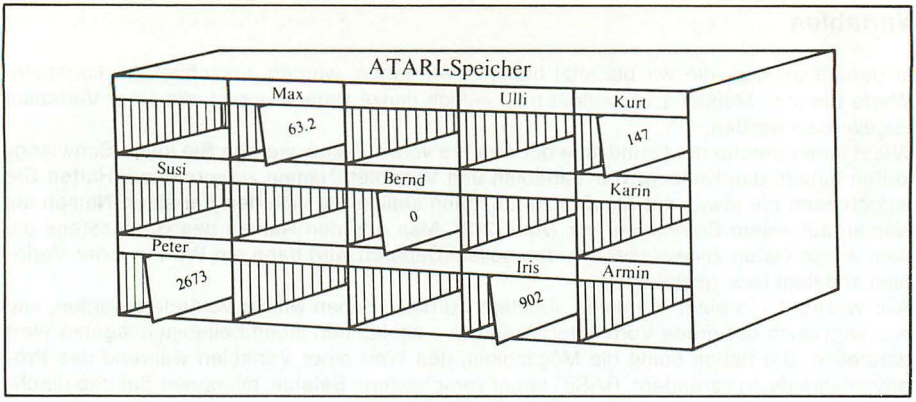


Abbildung 3-3: Variablen

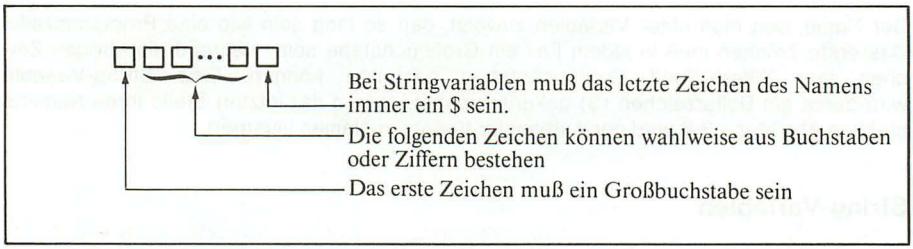


Abbildung 3-4: Variablennamen

Nachfolgend sehen Sie legale und illegale Namen für numerische Variablen:

Legal	Illegal
A	TEL-NUMMER
VORWAHL	3X4Z
X0	$SUMME
SPIELER	Punktzahl

Datenfelder

Als Datenfelder, auch Arrays genannt, bezeichnet man eine Anzahl mehrerer indizierter Variabler. Sie bilden einen wichtigen Bestandteil der meisten Computerprogramme. Wenn Sie sich im Moment noch nichts unter diesem Begriff vorstellen können, macht das gar nichts. Lesen Sie aufmerksam weiter, denn die folgenden Informationen sind sehr wichtig für Ihre Programmierfortschritte.
Im Prinzip ist ein Datenfeld eine ganz einfache Sache. Stellen Sie sich vor, Sie müßten zahlreiche Datenposten ablegen. Anstatt nun jedem von ihnen einen unterschiedlichen Variablennamen zuzuweisen, geben Sie der gesamten Datengruppe einen einzigen Namen. Diese Gruppe stellt das Datenfeld dar. Die Elemente, aus denen ein Datenfeld zusammengesetzt ist, sind numeriert. Man kann einzelne Daten daraus abfragen, indem man die jeweilige Positionsnummer angibt, die zuvor durch den Index zugewiesen wurde.

In ATARI-BASIC repräsentieren Datenfelder nur numerische Variablen. Kapitel 4 beschreibt, wie man Strings in Verbindung mit Datenfeldern behandelt.
Datenfelder bieten eine zeitsparende Lösung, um zahlreiche, zueinander in Verbindung stehende Variablen zu benennen. Stellen Sie sich vor, Sie hätten eine Tafel mit 200 Zahlen und müßten jeder dieser Zahlen einen unterschiedlichen Variablennamen zuteilen. Wäre es da nicht viel einfacher, der ganzen Tafel einen Namen zu geben und die einzelnen Zahlen mit ihrer Position auf der Tafel zu kennzeichnen? Genau das ist das Prinzip eines Datenfeldes.
Um Ihnen ein praktisches Beispiel zu demonstrieren, sollten Sie sich überlegen, welche Möglichkeiten sich bieten, um die einzelnen Punktzahlen in einem Bowling-Turnier zu verfolgen. Man könnte jedem Bowlingspieler einen separaten Variablennamen zuteilen (Abbildung 3-5). Das hätte den Vorteil, daß man die Variablen nach den Namen der Spieler benennen könnte. Aber was geschieht bei dem nächsten Turnier, bei dem die Spieler wieder andere Namen haben?
Hierfür würde sich anbieten, die Punktzahlen in einem Datenfeld abzulegen (Abbildung 3-6). Dadurch ist Ihr Programm auch völlig unabhängig von den Namen der Spieler. Wir nennen unser Datenfeld BOWLER. Jedes Element beinhaltet die Punktzahl eines Spielers. Dem Namen des Datenfeldes folgt der in Klammern eingeschlossene Index. Folglich wird also eine spezifische Datengruppe (das ist die Punktzahl eines Spielers) durch den Namen des Datenfeldes und den Index gekennzeichnet. Bei BOWLER(3) zum Beispiel, steht die Punktzahl des dritten Bowlingspielers.
Obgleich es in der Abbildung 3-6 nicht aufgezeigt ist, existiert in jedem Datenfeld ein Element mit dem Index 0. Deshalb gibt es zusätzlich zu BOWLER(1) bis BOWLER (10) auch noch einen BOWLER(0).

Ein- und zweidimensionale Datenfelder

Bevor Sie ein Datenfeld benutzen, müssen Sie dem Computer die maximale Anzahl der Elemente mitteilen, die dem Datenfeld zugewiesen werden sollen. Das wird mit Hilfe des DIM-Befehls durchgeführt, den wir aber erst später beschreiben werden.
Datenfelder in ATARI-BASIC werden mit einem oder zwei Indexen gekennzeichnet. Eindimensionale Datenfelder haben einen Index, während zweidimensionale mit zwei Indexen bezeichnet sind. Sie können sich ein eindimensionales Datenfeld wie eine Tafel vorstellen, auf der eine Zahlenreihe geschrieben steht (Abbildung 3-6). Durch den Index wird die Position der Zahlen innerhalb der Reihe bestimmt. Dagegen ist ein zweidimensionales Datenfeld wie eine Tafel, auf der Zahlen in Reihen und Spalten geschrieben stehen. Der erste Index gibt die Zeilenposition der Zahlen an, der zweite deren Spaltenposition.
Nun wollen wir unser Beispiel mit dem Bowling-Turnier erweitern. Stellen Sie sich vor, es gäbe fünf Teams, jedes mit zehn Spielern. Nun haben Sie vier Möglichkeiten, sich über die Punktzahlen der 50 Bowlingspieler auf dem Laufenden zu halten. Die erste Möglichkeit: Sie weisen jedem Spieler eine eigene Variable zu. Die zweite Möglichkeit: Sie teilen dem gesamten Turnier ein Datenfeld zu, das aus 50 verschiedenen Elementen besteht. Die dritte Möglichkeit: jedes Team erhält ein eigenes Datenfeld mit jeweils 10 Elementen. Die vierte und auch beste Möglichkeit ist, dem Turnier ein zweidimensionales Datenfeld zuzuweisen (Abbildung 3-7), wobei der erste Index die Teamnummer bezeichnet und der zweite die Nummer des Spielers. Bei BOWLER(3,2) steht also die Punktzahl des Spielers mit der Nummer 2 aus dem Team Nummer 3.

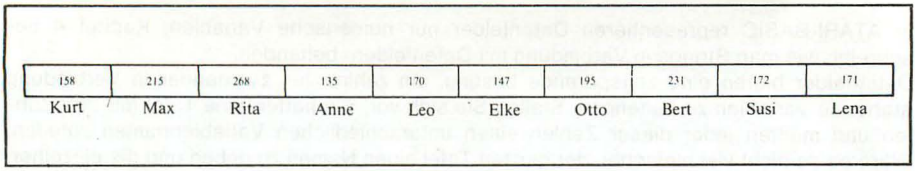


Abbildung 3-5: Anwendung verschiedener Variablennamen

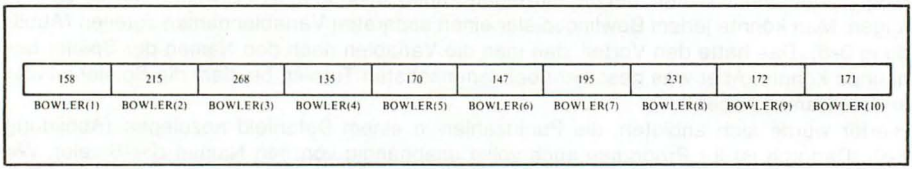


Abbildung 3-6: Anwendung eines Datenfeldes

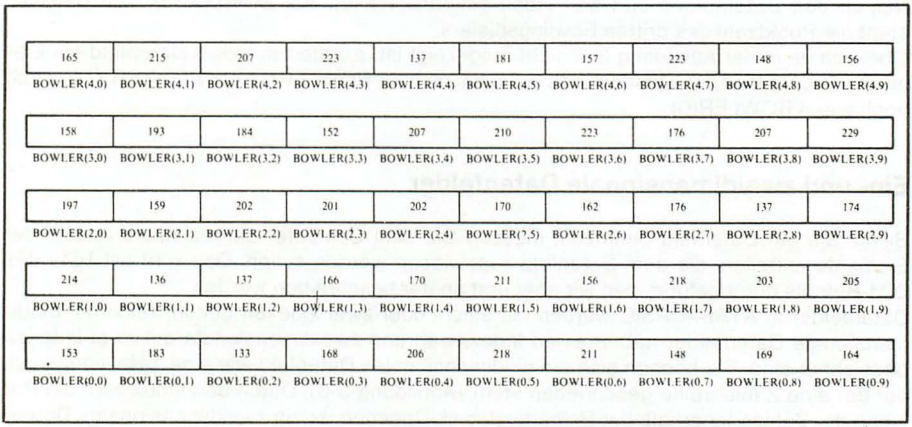


Abbildung 3-7: Anwendung eines zweidimensionalen Datenfeldes

Arithmetische Ausdrücke

Um die Werte von Variablen und Konstanten zu kombinieren, bedient man sich in der Programmiersprache arithmetischer Ausdrücke. Erinnern Sie sich, wie wir in der direkten Betriebsart einige einfache mathematische Aufgaben gelöst haben?
Der untenstehende Befehl weist den ATARI-Computer an, 4 und 6 zu addieren und die Summe darzustellen:

```
PRINT 4+6
10

READY
```

Dieser Befehl ist fast identisch:

```
PRINT A+B
0

READY
```

Er sagt dem Computer, die Werte der numerischen Variablen A und B zu addieren und die Summe daraus darzustellen.
Das Pluszeichen spezifiziert die Addition. In der Fachsprache spricht man von diesen Zeichen als Operatoren, wobei die Variable A und B die Operanden darstellen. Das Pluszeichen ist ein arithmetischer Operator, weil es eine Addition oder genauer gesagt eine arithmetische Operation, veranlaßt.
Arithmetische Operationen sind jedermann vertraut. Schon als Kind lernt man zu addieren, multiplizieren, subtrahieren und zu dividieren. Der Computer kennt jedoch noch andere Arten von Operatoren: Relations- und Boolesche Operatoren. Sie sind ebenso einfach zu verstehen, brauchen aber eine kurze Erläuterung, weil sie sich einer abstrakten Schreibweise bedienen.

Zusammengesetzte Ausdrücke

Der einfachste arithmetische Ausdruck besteht aus zwei Operanden und einem Operator. Man kann aber auch einfache Ausdrücke verknüpfen, um umfangreichere daraus zu bilden. Ebenso besteht die Möglichkeit, ganze Ausdrücke als Operanden in einem langen Ausdruck einsetzen. Diese können Sie anschließend wieder in einem anderen Ausdruck benutzen und so weiter. Derartige, sich aufeinander aufbauende Prozesse, nennt man zusammengesetzte Ausdrücke. Sie bilden einen wichtigen Bestandteil fast aller Programme.

Vorrang von Operatoren

Zusammengesetzte Ausdrücke werden nach bestimmten Regeln berechnet. In dem folgenden Beispiel soll eine Addition und Division in demselben Ausdruck durchgeführt werden:

```
PRINT A+B/10
```

Es gibt ein Standard-Schema, das die Reihenfolge der Berechnungen in einem Ausdruck festlegt. Diese Regeln gelten für numerische, Relations-, Boolesche und daraus kombinierte Ausdrücke. Zuerst aber werden wir einen Weg aufzeigen, der es ermöglicht, sich über die festgelegte Reihenfolge der Berechnungen hinwegzusetzen.

Umgehen der Vorrangregeln

Sie können die Reihenfolge, in der der ATARI-Computer die Ausdrücke berechnet, verändern, indem Sie Klammern benutzen. Jede Operation, die eingeklammert ist, wird zuerst ausgeführt. Wenn Sie in einem Ausdruck mehrere Klammersätze verwenden, beginnt der Computer mit der Berechnung ganz links und geht dann weiter zu den rechts liegenden Operationen.
Ein Klammerausdruck kann innerhalb eines anderen stehen. Das nennt man verschachtelte Klammern. Der ATARI-Computer beginnt mit der Berechnung bei dem innersten Klammerausdruck, danach fährt er mit der nächsthöheren Klammerebene fort und so weiter.
Sie können die Klammern unbegrenzt verschachteln, zum Beispiel auch um die Reihenfol-

ge der Operationen in einem Ausdruck zu erklären.
Hier sehen Sie einige arithmetische Berechnungen, bei denen Klammern verwendet wurden:

```
PRINT (2+10)*3
36

READY
PRINT (((2+10)*3)+31)*10
670

READY
PRINT -(2^(3+(8/4)))
-31.99993444

READY
```

Numerische Ausdrücke

Numerische Ausdrücke setzen sich aus numerischen Variablen und Konstanten zusammen. Sie enthalten arithmetische Operationen wie: (+) für Addition, (–) für Subtraktion, (*) für Multiplikation, (/) für Division und (^) für die Bildung einer Potenz. Das Minuszeichen (–) kennzeichnet außerdem einen negativen numerischen Wert. Die Operationen werden in folgender Reihenfolge ausgeführt: zuerst das negative Vorzeichen, danach die Potenzierung, anschließend Multiplikation oder Division und zum Schluß Addition und Subtraktion. Operationen derselben Vorrangstufe werden der Reihenfolge nach von links nach rechts berechnet.
Nachfolgend sehen Sie einige numerische Ausdrücke:

87.5 – 4.25	ergibt 79
1.5^(3/2/2/2)	ergibt 1.35540299
AL * (PL – 3.1 * CB)	ergibt: der Wert von AL multipliziert mit dem Ergebnis aus: 3.1 multipliziert mit dem Wert von CB, minus dem Wert von PL
7.5 * 2/5	ergibt 3

Relations-Ausdrücke

Relations-Ausdrücke bieten Ihnen die Möglichkeit, zwei Werte miteinander zu vergleichen um festzustellen, in welcher Beziehung sie zueinander stehen. Sie können vergleichen, ob der erste Wert größer, kleiner, ungleich, größer oder gleich oder aber kleiner oder gleich ist als der zweite Wert. Die Werte, die Sie auf diese Weise miteinander in Verbindung bringen, können Konstante, Variable oder alle Arten von arithmetischen Ausdrücken sein. Strings können Sie nur vergleichen, wenn auf beiden Seiten des Relations-Operators ein String steht.

Tabelle 3-1: Relations-Operatoren

Operation	Operator
kleiner als	<
größer als	>
gleich	=
ungleich	<>
größer als oder gleich	>=
kleiner als oder gleich	<=

Ist die gestellte Bedingung wahr, so erhält der Relations-Ausdruck den numerischen Wert 1. Ist die gestellte Bedingung jedoch falsch, so ergibt der Relations-Ausdruck den numerischen Wert 0.
In der Tabelle 3-1 sind sämtliche Relations-Operatoren aufgeführt. Sie haben alle dieselbe Vorrangstufe. Werden mehrere Operatoren dieser Art in einem Ausdruck verwendet, so werden sie der Reihenfolge nach von links nach rechts berechnet.
Hier sehen Sie ein paar Beispiele für Relations-Ausdrücke:

1 = 5 – 4	ergibt 1 (wahr)
14 > 66	ergibt 0 (falsch)
15 >= 15	ergibt 1 (wahr)
"AA" > "AA"	ergibt 0 (falsch)
(A = B) = (A$ > B$)	das Ergebnis hängt vom Wert der Variablen ab. Wenn der Wert A gleich dem Wert von B ist und der Wert von A$ größer als der Wert von B$ ist, dann ist die Bedingung wahr und ergibt 1 (wahr).

Relations-Ausdrücke sind eigentlich ganz einfach zu verstehen. Es wird jedoch komplizierter, wenn diese als ein Teil eines numerischen Ausdrucks eingesetzt werden. Die Werte der Relations-Ausdrücke (0 für falsch und 1 für wahr) sind legitime numerische Werte. Das klingt jetzt vielleicht etwas verwirrend. Welchen Sinn ergibt zum Beispiel der Ausdruck (9 = 9) * 4? Keinen. Es sei denn, er ist Bestandteil eines BASIC-Programms. In BASIC ist wahr gleich 1, und da (9 = 9) wahr ist, bedeutet der Ausdruck (9 = 9) * 4 das gleiche wie 1 * 4. Sehen Sie sich die Beispiele an:

25 + (14 > 66)	bedeutet 25 + 0
A + (1 = B – 4)	bedeutet A + 1 wenn B = 5, oder A + 0 wenn die Bedingung falsch ist.

Strings vergleichen

Wahrscheinlich haben Sie sich schon gefragt, nach welchem Prinzip der ATARI-Computer arbeitet, wenn er Strings vergleicht. Er prüft dabei nach zwei Gesichtspunkten. Der erste betrifft die Länge des Strings. Strings, die unterschiedlich lang sind, können nie den gleichen Wert haben (denken Sie daran, daß die Leerzeichen bei der Stringlänge ebenso zäh-

len). Obwohl zum Beispiel der erste Teil eines langen und eines kurzen Strings übereinstimmen, so ist doch der lange String größer als der kurze.
Die zweite Regel, nach der der Computer vorgeht, ist zu prüfen, ob die Strings die gleichen Zeichen in der gleichen Reihenfolge enthalten. Dazu wird der Vergleich bei den jeweils ersten Zeichen begonnen, anschließend wird mit den jeweils zweiten Zeichen fortgefahren und so weiter. Bis entweder einer der beiden Strings keine zu prüfende Zeichen mehr enthält oder der Computer ein Zeichen gefunden hat, an dem sich die Strings unterscheiden. Die Rangfolge, durch die der Computer die Buchstaben einordnen kann, lautet: A < B, B < C, C < D usw. Zahlen, die Teil eines Strings sind, werden nach der normalen Reihenfolge sortiert: 0 < 1, 1 < 2, 2 < 3 usw. Bei Sonderzeichen, innerhalb von Strings, wie zum Beispiel +, – oder $, verfährt der Computer gemäß der Liste in Anhang D.

Boolesche Ausdrücke

Mit Hilfe Boolescher Operatoren kann ein Programm logische Operationen ausführen. Deshalb werden sie auch logische Operatoren genannt. Es gibt vier Boolesche Operatoren: AND, OR, Exclusive OR und NOT. ATARI-BASIC kennt drei dieser Operatoren: AND bedeutet logisches Und, OR bedeutet logisches Oder und NOT bedeutet eine logische Umkehrung bzw. eine Negation.
Wenn Sie den Sinn dieser neuen Ausdrücke noch nicht verstanden haben, dann hilft Ihnen vielleicht das folgende Supermarkt-Beispiel weiter.
Stellen Sie sich vor, Sie gingen mit Ihren beiden Kindern, Iris und Walter, Kuchen einkaufen. Der AND-Operator sagt Ihnen, den Kuchen zu kaufen, den beide Kinder gewählt haben. Der OR-Operator sagt Ihnen, den Kuchen zu nehmen, den eines der beiden Kinder ausgesucht hat. Der NOT-Operator stellt immer das Gegenteil dar. Wenn Iris sich weigert, mit Walter übereinzustimmen, dann ist die Entscheidung von Iris immer das NOT von Walters Entscheidung.
Da Computer mit Übereinstimmungen oder Ähnlichkeiten nichts anfangen können, werden die Resultate Boolescher Operatoren in Zahlen berechnet, wobei wieder wahr = 1 und falsch = 0 bedeutet. Da Relations-Operatoren ihre Ergebnisse auch in den Werten 0 und 1 angeben, werden diese oft in Verbindung mit Booleschen Operatoren verwendet. Im nächsten Abschnitt erfahren Sie, wie man Boolesche Operatoren mit anderen Operanden in Verbindung bringt.
Tabelle 3-2 zeigt zusammengefaßt, wie Boolesche Ausdrücke berechnet werden. Boolesche Operatoren haben alle dieselbe Vorrangstufe. Wenn mehrere dieser Operatoren in einem Ausdruck vorkommen, so werden sie in der Reihenfolge von links nach rechts bearbeitet.
Hier sehen sie Beispiele mit Booleschen Ausdrücken:

NOT ((3 + 4) > 6)	ergibt 0 (falsch)
(''AA'' = ''AB'') OR ((8 * 2) = 4↑2)	ergibt 1 (wahr)
NOT (''APFEL'' = ''BIRNE'') AND (A$ = B$)	ergibt 1 (wahr), wenn A$ und B$ gleich sind; andernfalls 0 (falsch).

Tabelle 3-2: Boolesche Wahrheits-Tabelle

Die AND-Operation ergibt 1, wenn beide Werte 1 betragen.

1 AND 1 = 1	1 AND 0 = 0
0 AND 1 = 0	0 AND 0 = 0

Die OR-Operation ergibt 1, wenn einer der Werte 1 beträgt.

1 OR 1 = 1	1 OR 0 = 1
0 OR 1 = 1	0 OR 0 = 0

Die NOT-Operation führt eine logische Vertauschung von 1 (wahr) und 0 (falsch) durch.

NOT 1 = 0
NOT 0 = 1

Ausdrücke mit unterschiedlichen Operatoren

Ist es möglich, daß man einen Operator nicht mit bestimmten Operanden kombinieren kann? Natürlich. Man kann zum Beispiel String-Operanden nicht zusammen mit booleschen oder numerischen Operatoren benutzen, sondern nur mit Relations-Operatoren. Das heißt, daß sich Strings ausschließlich mit anderen Strings vergleichen lassen. Die übrigen Operanden können Sie nebeneinander in einem Ausdruck verwenden.
Der ATARI-Computer muß, wenn er einen Ausdruck mit unterschiedlichen Operatoren berechnet, verschiedene Fragen klären. Zum Beispiel, welche Operation zuerst ausgeführt wird.
In der Tabelle 3-3 sind alle Operatoren von der höchsten bis zur niedrigsten Vorrangstufe aufgeführt. Die Liste zeigt außerdem, daß zuerst die eingeklammerten Angaben berechnet werden. Wurden in einem Ausdruck mehrere Klammerpaare angewendet, dann beginnt der ATARI-Computer mit der Auswertung bei dem innersten Klammerausdruck, fährt dann mit dem zweit-innersten fort usw. (Sie erinnern sich sicher an das Prinzip der Klammerverschachtelung). Anschließend werden die numerischen Ausdrücke, danach die Relations-Ausdrücke und zum Schluß die Booleschen Ausdrücke berechnet.
Wie Sie bereits wissen, verwandelt ATARI-BASIC Boolesche und Relations-Ausdrücke in die Werte 0 für falsch und 1 für wahr. Umgekehrt wird der numerische Wert 0 in falsch, während alle anderen numerischen Werte in wahr umgewandelt werden. Strings können nicht in Zahlenwerte verwandelt werden. Das Ergebnis eines String-Vergleichs wird jedoch auch in numerischen Werten (0 oder 1) ausgegeben, die Sie dann wiederum in einen arithmetischen Ausdruck einfügen können.
Hier sind einige Beispiele für Ausdrücke mit unterschiedlichen Operatoren:

Legal	**Illegal**
43 AND 37	160 + "HEGARSTR"
(A$ = B$) AND – 6.25	ST$ <= A
(K = M) = (Z$ = Y$)	NOT (A$) = B$

Tabelle 3-3: Operatoren

	Vorrang-stufe	Operator	Bedeutung
	hoch 9	( )	Die Klammern bestimmen die Reihenfolge der Berechnungen
arithmetischer Operator	8	^	Potenzierung
	7	–	negatives Vorzeichen
	6	*	Multiplikation
	6	/	Division
	5	+	Addition
	5	–	Subtraktion
Relations-Operatoren	4	=	Gleich
	4	< >	Ungleich
	4	<	Kleiner als
	4	>	Größer als
	4	< =	Kleiner als oder gleich
	4	> =	Größer als oder gleich
Boolesche Operatoren	3	NOT	Logische Vertauschung
	2	AND	Logisches Und
	1	OR	Logisches Oder

Schlüsselworte

Alle Worte, die BASIC-Anweisungen oder auch nur Bestandteile daraus darstellen, nennt man Schlüssel- oder auch geschützte BASIC-Worte. In Anhang E sind alle Schlüsselworte, die ATARI-BASIC kennt, aufgeführt. Sie werden in diesem Kapitel noch auf einige dieser geschützten Worte stoßen; die übrigen lernen Sie im Verlauf des Buches kennen.
Bei der Programmausführung durchsucht der ATARI-Computer jede BASIC-Anweisung nach Schlüsselworten. Zu Schwierigkeiten kann es kommen, wenn er ein Schlüsselwort in einem Variablennamen findet und dessen Bedeutung falsch interpretiert. Deshalb erleichtern Sie Ihrem Computer die Arbeit und vermeiden Sie die Anwendung von geschützten BASIC-Worten in Variablennamen.

Abgekürzte Schlüsselworte

Schon am Anfang des Buches haben Sie gelernt, daß man den PRINT-Befehl durch ein Fragezeichen abkürzen kann. ATARI-BASIC erlaubt Ihnen, die meisten Anweisungen zu verkürzen. Oft genügt es, wenn der erste Buchstabe des Befehls eingegeben wird. Zum Beispiel steht das L. für LIST. Um dem Computer die Überlegung abzunehmen, ob es sich bei diesem L um eine Variable oder um ein abgekürztes Schlüsselwort handelt, muß hinter jeden verkürzten Befehl ein Punkt gesetzt werden; zum Beispiel L. für LIST.

In vielen Fällen genügt jedoch ein einzelner Buchstabe nicht. Das L könnte der Computer als LIST oder LOAD interpretieren. Wenn also verschiedene Befehle mit demselben Buchstaben beginnen, benötigt der Computer mehrere Buchstaben, um die Anweisung exakt zu spezifizieren und Mißverständnisse auszuschließen.
In Anhang E finden Sie eine vollständige Liste der abgekürzten Schlüsselworte. Sie beinhaltet die jeweils kürzeste Version, wobei es Ihnen überlassen bleibt, ob Sie die noch folgenden Buchstaben des Befehlswortes zufügen möchten. Zum Beispiel akzeptiert der Computer anstatt der ausgeschriebenen Anweisung PRINT auch PRIN., PRI., PR. oder?.

BASIC-Anweisungen

Nun kommen wir zu dem dritten Element der BASIC-Syntax: den Anweisungen. Jede dieser Anweisungen weist den ATARI-Computer an, eine bestimmte Operation auszuführen oder vorher definierte Schritte zu unternehmen.
Dieses Kapitel macht Sie mit dem Konzept des Programmierens vertraut, wobei Sie anhand von praktischen Beispielen lernen, die einzelnen Befehle sinnvoll zu gebrauchen. In Kapitel 11 können Sie zwischendurch die exakten Definitionen und Feinheiten der Anweisungen nachlesen.
Eine Bemerkung noch vorweg: Dieses Buch beschäftigt sich mit den Grundlagen der Programmerstellung, es kann jedoch nicht das gesamte Sachgebiet des Programmierens erläutern. Wenn Sie mehr Informationen über dieses Thema sammeln wollen, dann schlagen Sie in Anhang I nach, wo Sie eine Liste weiterer Fachliteratur finden.

Erläuterungen innerhalb eines Programms

Wir beginnen nun die Beschreibung der BASIC-Befehle mit der einzigen Anweisung, die der Computer nicht ausführt: der REMARK-Anweisung (abgekürzt REM). Haben Sie am Anfang einer Programmzeile den Befehl REM eingegeben, so ignoriert der Computer diese Zeile völlig und springt weiter zur nächsten Zeile. Warum also benutzt man eine solche Anweisung? Die Antwort ist ganz einfach: Um in Programmen Erläuterungen einzufügen und sie dadurch leichter verständlich zu gestalten.
Wenn Sie ein kurzes Programm geschrieben haben, das nur fünf oder zehn Befehle enthält, werden Sie wahrscheinlich keine Schwierigkeiten haben, sich an den Inhalt dieses Programms zu erinnern. Es sei denn, Sie benutzen es für längere Zeit nicht und haben vergessen, was die Anweisungen bewirken. Sobald Sie jedoch ein Programm mit 100 oder 200 Befehlen schreiben, wird es sehr schwierig, sich alle wichtigen Einzelheiten zu merken. Nachdem Sie dann zahlreiche Programme entwickelt haben, wird es unmöglich, sich an jedes Detail zu erinnern. Deshalb ist es von Vorteil, wenn Sie mit Hilfe der Anweisung REM die einzelnen Programmteile kommentieren.
Erfahrene Programmierer machen von der Möglichkeit, mittels REM wichtige Erklärungen im Programm unterzubringen, häufig Gebrauch. Auch wir benutzen diesen Befehl in allen unseren Beispielen, um Sie damit vertraut zu machen und um Sie zu motivieren, diesen Stil für Ihre eigenen Programme zu kopieren.
Eine Programmzeile, die die Anweisung REM enthält, wird, wie jede andere Zeile auch, mit einer Zeilennummer begonnen.

LET

Mit der Anweisung LET definiert man eine Variable, das heißt, man weist ihr einen bestimmten Wert zu. Anweisungen dieser Art bilden einen wichtigen Bestandteil der BASIC-Programme. Dazu ein Beispiel:

```
90 REM DEFINITION DER VARIABLEN X
100 LET X=3
```

In der Programmzeile 100 wird der Variablen X der Wert 3 zugewiesen. Die gleiche Anweisung kann man auch verkürzen:

```
100 X=3
```

Das Wort LET kann weggelassen werden.
Im nächsten Beispiel wird eine String-Variable definiert:

```
215 A$="MUENCHEN"
```

Der String-Variablen wurde das Wort MUENCHEN zugewiesen. Die Anführungsstriche kennzeichnen den String, bilden jedoch keinen Bestandteil seines Inhalts.
Nachfolgend sehen Sie, wie wir den schon bekannten BOWLER(1) bis BOWLER(3) Werte zuordnen:

```
200 REM BOWLER() HAT DEN PUNKTSTAND!
210 BOWLER(1)=150
220 BOWLER(2)=210
230 BOWLER(3)=268
```

Wie Sie schon wissen, kann man mehrere Befehle in eine Zeile schreiben. Deshalb könnte die Definition der drei Variablen auch so aussehen:

```
200 REM BOWLER() HAT DEN PUNKTSTAND!
210 BOWLER(1)=150:BOWLER(2)=210:BOWLER(3)=268
```

Denken Sie daran, daß die Anweisungen durch einen Doppelpunkt voneinander getrennt werden müssen.
Zuweisungen können auch alle arithmetischen und logischen Operationen beinhalten, die Sie vorher schon kennengelernt haben.

```
100 V=33+7/9
```

Dieser Befehl weist der Variablen V den Wert 33.77777777 zu; die folgenden drei Programmzeilen bewirken dasselbe:

```
90 REM X+Y WERDEN DEFINIERT
100 X=7
110 Y=9
120 V=33+X/Y
```

Die drei Befehle können Sie auch zu einer Programmzeile umformen:

```
100 X=7:Y=9:V=33+X/Y
```

Und nun zwei Definitionen, in denen Boolesche Operationen verwendet werden:

```
90 REM DEFINITIONEN MIT BOOLESCHEN OPERATIONEN
100 A=NOT((3+4)=6)
110 B=("AA"="AB") OR ((8*2)=(4^2))
```

DATA und READ

Eine bequeme Art, mehreren Variablen Daten zuzuweisen, bietet Ihnen ATARI-BASIC mit dem DATA- und READ-Befehl.
Dazu folgendes Beispiel:

```
5 REM DEFINIERE ALLE VARIABLEN
10 DATA 10,20,-4,300
20 READ A,B,C,D
```

Die Anweisung in Zeile 10 enthält vier numerische Werte, die den vier numerischen Variablen aus Zeile 20 zugeordnet werden. Nachdem die Programmzeile 20 ausgeführt wurde, ist A = 10, B = 20, C = – 4 und D = 300.
Alle Daten aus Ihrem Programm, die in DATA-Anweisungen stehen, können Sie sich als eine Liste vorstellen (Abbildung 3-8), wobei ein DATA-Befehl mit 10 Werten die gleich Liste bildet wie zwei DATA-Befehle mit jeweils 5 Werten.
Der Computer bedient sich zum Markieren der letzten gelesenen Angabe eines READ/-DATA-Zeigers. Die READ-Anweisung liest die Werte der Reihenfolge nach aus der DATA-Liste und weist diese den READ-Variablen zu. Der Computer merkt sich dabei, welche Werte er gelesen hat, indem er den READ/DATA-Zeiger jeweils um eine Stelle weiterrückt (Abbildung 3-9).
Eine DATA-Liste kann aus Strings und numerischen Werten bestehen. Bitte beachten Sie dabei, daß die Werte konstant sein müssen, das heißt, daß in einer DATA-Zeile keine arithmetischen Ausdrücke erlaubt sind. Dagegen können Konstante auch in der Exponential-Schreibweise angegeben werden. Der ATARI-Computer nimmt Strings nur in der DATA-Liste auf, wenn diese ohne Anführungsstriche eingetippt werden.
Beachten Sie die beiden DATA-Zeilen:

```
10 DATA 0,-0.8,1.2558E3
20 DATA A+B, SUMME,NAME$
```

Zeile 10 beinhaltet drei numerische Werte: 0, –0,8 und 1.2558E3. Dagegen sehen Sie in der Programmzeile 20 drei Strings. Der erste Wert besteht aus drei Zeichen und der zweite aus sechs, da das Leerzeichen vor dem Wort SUMME als Teil des Strings gewertet wird.

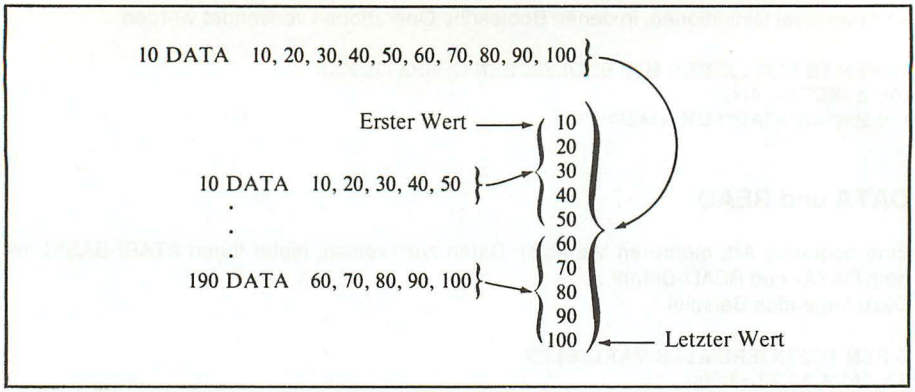


Abbildung 3-8: Die Werte der DATA-Anweisung bilden eine Liste

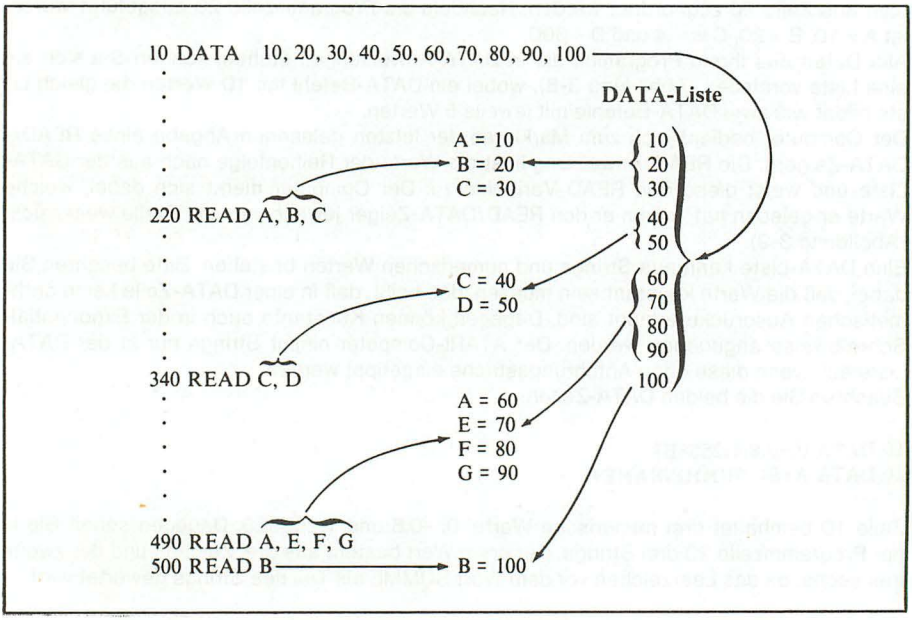


Abbildung 3-9: Der READ-Befehl liest die Werte der DATA-Liste der Reihenfolge nach heraus

Wie Sie schon wissen, können Sie in diesen Anweisungen mit Strings und numerischen Werten arbeiten. Dabei ist darauf zu achten, daß die Variablenart hinter READ immer den zu lesenden Werten entspricht. Numerische Werte können also nur numerischen Variablen und Strings nur String-Variablen zugewiesen werden. Der ATARI-Computer gibt eine Fehlermeldung aus, wenn Sie versuchen, die verschiedenen Variablentypen untereinander zu vermischen.

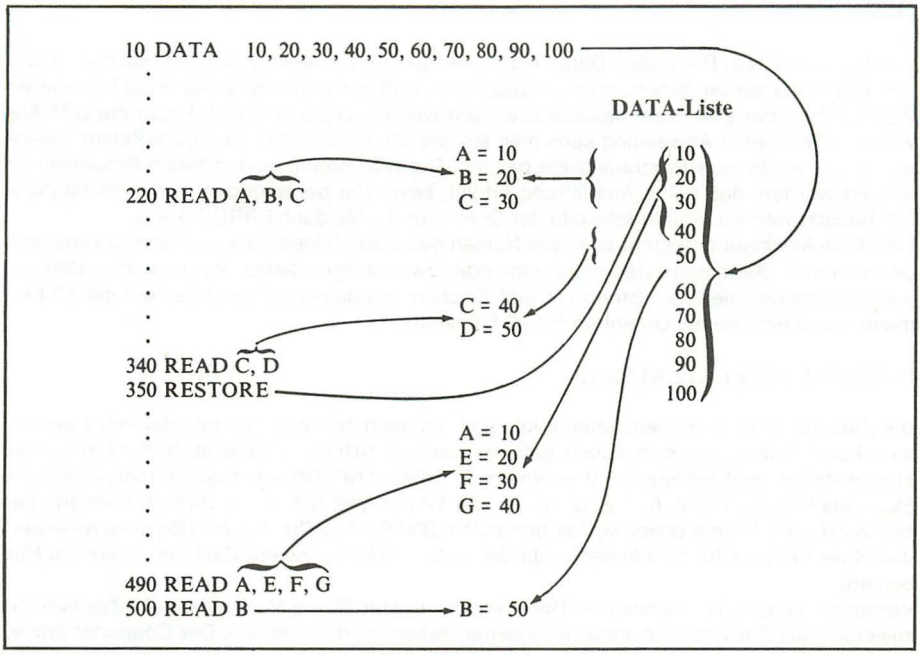


Abbildung 3-10: Der RESTORE Befehl bewirkt, daß der READ/DATA-Zeiger wieder auf den Anfang der DATA-Liste gestellt wird.

RESTORE

Mit dem Befehl RESTORE veranlassen Sie den Computer, den READ/DATA-Zeiger zurück zum Anfang der DATA-Liste zu setzen (Abbildung 3-10). Dadurch wird beim nächsten Lesevorgang wieder der erste Wert aus der DATA-Liste entnommen.
Sehen Sie dazu folgendes Beispiel:

```
10 DATA 1,2,3,4,5
20 DATA 10,20,30,40,50
30 READ A
40 PRINT A
49 REM RUECKE DATA-ZEIGER AUF ZEILE 20
50 RESTORE 20
60 READ A
70 PRINT A
```

Die Programmzeile 30 sagt aus, den ersten Wert der DATA-Liste der Variablen A zuzuweisen. RESTORE 20 bewirkt, daß der READ/DATA-Zeiger an den Anfang der Daten in Zeile 20 gesetzt wird. Somit liest READ (in Zeile 60) den ersten Wert der DATA-Liste (Zeile 20) und weist ihn der Variablen A zu.

DIM

Wenn Sie in Ihrem Programm Datenfelder oder String-Variablen benutzen möchten, müssen Sie diese vorher dimensionieren. Das heißt, daß Sie dem Computer mitteilen, wieviel Zeichen Sie den Variablen maximal zuweisen werden. Dazu verwendet man die DIM-Anweisung. Mit dieser Anweisung kann man soviele String-Variablen und Datenfelder dimensionieren, wie in eine Programmzeile passen. Der DIM-Befehl muß in Ihrem Programm so plaziert werden, daß seine Ausführung erfolgt, bevor die betreffenden Variablen tatsächlich benutzt werden. Andernfalls gibt der Computer die Meldung ERROR 9 aus.
Eine DIM-Anweisung besteht aus dem Namen des Datenfeldes, bzw. der String-Variablen, und einem in Klammern stehenden ein- oder zweistelligen Index. Im folgenden Beispiel werden dimensioniert: ein String mit fünf Zeichen, ein numerisches Datenfeld mit 13 Elementen und ein zweites Datenfeld mit 20 Elementen:

```
10 DIM S1$(5),NB(12),BOWLER(1,9)
```

Die Zahl, die dem Variablennamen folgt, gibt den höchsten Wert an, den der Index annehmen kann. Haben Sie sich schon gefragt, weshalb NB(12) 13 Elemente hat? Nun, das kommt daher, weil Indizes bei 0 anfangen zu zählen. NB(12) stellt dementsprechend die Elemente 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 und 12 dar. Wie Sie sich sicherlich erinnern, haben wir dieses Thema schon vorher behandelt. DIM BOWLER(8,10) zum Beispiel reserviert den Speicherplatz für 99 Elemente, da die erste sowie die zweite Zahl des Index bei Null beginnt.
Versuche, einem dimensionierten Datenfeld oder einer String-Variablen mehr Zeichen zuzuweisen, als Sie vorher im Index angegeben haben, sind vergeblich. Der Computer ordnet die ersten Zeichen bis zur Indexgrenze zu, die restlichen Zeichen werden ignoriert.

CLR

Die Dimensionierung von Datenfeldern und String-Variablen kann, wenn sie einmal durchgeführt wurde, während des Programmablaufes nicht mehr verändert werden. In ATARI-BASIC haben Sie jedoch die Möglichkeit, alle vorher dimensionierten Variablen mit dem Befehl CLR zu löschen. Ebenso veranlaßt diese Anweisung, alle numerischen Variablen auf Null zu setzen. Ferner wird der READ/DATA-Zeiger, genau wie RESTORE, an den Anfang der DATA-Liste gerückt. Sehen Sie dazu die folgenden Beispiele:

```
X=37

READY
PRINT X
37

READY
10 DATA 1,2,3,4,5
READ A,B

READY
PRINT A,B
1        2

READY
CLR

READY
PRINT X
0
```

```
READY
READ A,B

READY
PRINT A,B
1       2

READY
```

Programmverzweigungen

Die Anweisungen eines BASIC-Programms werden im Normalfall gemäß der Reihenfolge der Zeilennummern ausgeführt (Abbildung 3-1). Mit Verzweigungsbefehlen kann man diese Reihenfolge jedoch verändern.

GOTO

Die einfachste Verzweigungsanweisung ist GOTO. In diesem Befehl gibt man die Zeilennummer an, die als nächstes ausgeführt werden soll und führt somit einen "unbedingten Sprung" durch. Dazu das folgende Beispiel:

```
20 A=4
30 GOTO 100
40 B=5
50 C=6
100 PRINT A
110 PRINT B*C
```

Der Befehl in Zeile 20 weist der Variablen A den Wert 4 zu. Als nächstes folgt der GOTO-Befehl, der aussagt, daß mit der Programmdurchführung in Zeile 100 fortgefahren werden soll. Durch diesen Verzweigungsbefehl wird die Ausführung nach Zeile 30 unterbrochen und in Zeile 100 fortgesetzt. Damit die Anweisungen in Zeile 40 und 50 auch bearbeitet werden, muß im nachfolgenden Programm wieder auf diese Zeilen zurückverzweigt werden.
Sie können in Ihrem Programm Verzweigungen zu jeder Zeilennummer einfügen, selbst wenn in der betreffenden Zeile nur die Anweisung REM steht. Wie Sie schon wissen, ignoriert der Computer die Bemerkungen und fährt mit der Ausführung in der nächsten Zeile fort. Sehen Sie dazu folgendes Programm:

```
20 A=4
30 GOTO 70
40 B=5
70 REM BEINHALTET NUR ERLAEUTERUNGEN
80 PRINT A,B
```

Der Computer bricht die Programmausführung nach Zeile 30 ab und verzweigt zu Zeile 70. Da er in dieser Zeile jedoch nur den für ihn unwichtigen Befehl REM findet, setzt er mit der Ausführung erst wieder in Programmzeile 80 ein. Man hätte also auch gleich zu Zeile 80 verzweigen können:

```
20 A=4
30 GOTO 80
40 B=5
70 REM BEINHALTET NUR ERLAEUTERUNGEN
80 PRINT A,B
```

Der ATARI-Computer akzeptiert ebenso, wenn Sie die Verzweigungsadresse in Form eines arithmetischen Ausdrucks angeben:

```
10 DATA 0,1,2,3,4
20 READ A
30 PRINT A
40 GOTO 30*A+50
50 PRINT "ZEILE 50"
60 GOTO 20
80 PRINT "ZEILE 80"
90 GOTO 20
110 PRINT "ZEILE 110"
120 RESTORE
130 GOTO 20
```

Bevor das Programm zu einer bestimmten Zeile verzweigen kann, muß der Computer den arithmetischen Ausdruck in Zeile 40 berechnen. Wenn die Variable A = 0 ist, erfolgt die Programmfortführung in Zeile 50, wenn A = 1 ist, in Zeile 80 und schließlich wenn A = 2 ist, in Zeile 110.
Sollten Sie versuchen, zu einer Zeilennummer zu verzweigen, die in dem Programm nicht existiert, gibt der Computer eine Fehlermeldung aus.
Nun tippen Sie das folgende Programm ein und starten es anschließend mit RUN.

```
9 REM DEFINITION DER VARIABLEN B
10 B=4
20 PRINT B
30 A=B-3
40 GOTO 30*A+50
50 END
79 REM B=4
80 PRINT B
90 B=5
100 GOTO 20
109 REM B=5
110 PRINT B
120 B=3
130 GOTO 20
```

Versuchen Sie, den Programmablauf nachzuvollziehen, und ändern Sie das Programm so um, daß es die Ziffern 3, 4, 5, 3, 4, 5, 3, 4, 5 ..... in laufender Reihenfolge darstellt.

ON/GOTO

ATARI-BASIC kennt noch eine andere Version des GOTO-Befehls. Das folgende Programm veranschaulicht, wie dieser eingesetzt wird:

```
10 DATA 0,1,2,3,4
20 READ A
30 PRINT A
40 ON A+1 GOTO 50,80,110
50 PRINT "ZEILE 50"
60 GOTO 20
80 PRINT "ZEILE 80"
90 GOTO 20
110 PRINT "ZEILE 110"
120 RESTORE
130 GOTO 20
```

In Zeile 40 sehen Sie den Befehl ON/GOTO, der einen "bedingten Sprung" bewirkt. Wenn der Wert der Variablen A = 0 ist, verzweigt das Programm zu Zeile 50. Ist A = 1, geht es zu Zeile 80, und schließlich bei A = 2 wird das Programm in Zeile 110 weitergeführt.
Der ON/GOTO-Befehl in Zeile 40 enthält einen numerischen Ausdruck und mehrere Zeilennummern. Zuerst berechnet der ATARI-Computer den Ausdruck, der zwischen ON und GOTO steht. Wenn das Ergebnis 1 ist, verzweigt das Programm zu der jeweils ersten Adresse nach GOTO. Bei Wert 2 wird ein Sprung zu der zweiten Zeilenangabe durchgeführt usw. Ist das Ergebnis des Ausdrucks gleich 0 oder aber größer als die Anzahl der aufgeführten Zeilennummern hinter GOTO, wird keine Programmverzweigung durchgeführt und mit dem nächsten Befehl nach ON/GOTO fortgefahren.
Ergibt das Ergebnis des Ausdrucks einen negativen oder einen Wert größer als 255, führt dies zu einer Fehlermeldung.
Die Zeilennummern kann man als Konstante oder auch als numerische Variablen angeben. Das untenstehende ATARI-BASIC-Programm demonstriert Ihnen die Funktion des ON/GOTO-Befehls:

```
10 B=4
20 PRINT B
30 A=B-2
40 ON A GOTO 180,70,150
49 REM WERT IST GLEICH 0 ODER ZU GROSS
50 PRINT "NOCH EINMAL"
60 GOTO 10
69 REM B=4
70 PRINT B
80 B=5
90 GOTO 30
149 REM B=5
150 PRINT B
160 B=3
170 GOTO 20
179 REM B=1
180 PRINT B
190 B=6
200 GOTO 30
```

Programmschleifen

Die Befehle GOTO und ON/GOTO ermöglichen Ihnen, die Reihenfolge der Anweisungsausführungen – gemäß der Programmlogik – festzulegen. Oftmals ist es jedoch nötig, bestimmte Programmabschnitte wiederholt ausführen zu lassen. Stellen Sie sich zum Beispiel vor, jedem der 100 Elemente des Datenfeldes A( ) muß ein Wert zugewiesen werden. Man könnte jetzt 100 LET-Anweisungen schreiben, aber das wäre doch reichlich mühsam. Selbst mit der Anwendung von READ/DATA würde dieser Vorgang noch zu viel Zeit in Anspruch nehmen. Zum Glück bietet Ihnen ATARI-BASIC eine schnelle Lösung für solche Probleme in Form von einer Programmschleife an.

FOR-NEXT

Wir zeigen Ihnen, wie man auf einfachste Weise die 100 Elemente des Datenfeldes A indiziert:

```
10 DIM A(99)
20 FOR N=0 TO 99 STEP 1
30 A(N)=1
40 NEXT N
```

Die Anweisungen, die zwischen FOR und NEXT stehen, werden wiederholt ausgeführt. In dem oben gezeigten Programm sehen Sie nur einen Zuweisungsbefehl, der sooft ausgeführt wird, wie in der FOR-NEXT-Schleife festgelegt wurde.
Damit Sie gleich sehen, wie diese Programmschleife funktioniert, haben wir noch ein Beispiel für Sie:

```
10 DIM A(99)
20 FOR N=0 TO 99 STEP 1
30 A(N)=N
35 PRINT A(N)
40 NEXT N
50 END
```

Schreiben Sie nun RUN. Anschließend können Sie beobachten, wie den Variablen bei jedem Schleifendurchlauf die einzelnen Werte zugewiesen werden. Insgesamt sind es 100 Zahlen; von 0 bis 99.
Die Anweisungen, die zwischen FOR und NEXT stehen, werden sooft ausgeführt, wie in dem Index rechts neben FOR angegeben wurde. In unserem Beispiel heißt der Index N und veranlaßt, daß die Schleife 100 mal in Einer-Schritten (STEP 1) durchlaufen wird. Die Variable N erscheint aber auch noch in Zeile 30. Beim ersten Schleifendurchlauf nimmt N den Wert 0 an. Somit sieht die erste Zuweisung folgendermaßen aus:

```
30 A(0)=0
```

Am Anfang ist N gleich 0, erhöht sich aber bei jedem Durchlauf um den Wert, der in Zeile 20 bei STEP angegeben wurde (in unserem Beispiel ist das 1). Nach dem zweiten Schleifendurchlauf hat A( ) also folgenden Wert:

```
30 A(1)=1
```

Die Index-Variable N wird nun fortlaufend um eins erhöht, bis sie den Wert 99 erreicht hat. Der Anfangs- und Endwert sowie die Zählschritte (STEP) können Zahlen, numerische Variablen oder arithmetische Ausdrücke sein. Ändern Sie nun in Zeile 20 die Größe des Zähl-

schrittes in 5 um, und lassen Sie das Programm ablaufen. Wie Sie sehen, wird die Zuordnungsanweisung in Zeile 30 nur noch 20 mal ausgeführt, und der Endwert von N ist jetzt 95. Verändern Sie das Programm so, daß der maximale Wert der Schleifenvariablen N nun 500 ist. Vergessen Sie dabei jedoch nicht, auch die Dimensionierung von A zu erweitern.
Der Zählschritt kann eine positive oder negative Angabe sein. Wenn Sie eine negative Zahl verwenden, muß natürlich der Anfangswert von N größer sein als sein Endwert. Wie könnte man das vorhergehende Programmbeispiel so abändern, daß man das gleiche Ergebnis bei der Angabe STEP –1 erhält?

```
10 DIM A(99)
20 FOR N=99 TO 0 STEP -1
30 A(N)=N
35 PRINT A(N)
40 NEXT N
50 END
```

Lassen Sie nun das Programm ablaufen und beobachten Sie, wie der Computer mit dem negativen Zählschritt rechnet.
ATARI-BASIC bietet ihnen eine weitere Erleichterung an: Die Angabe STEP darf ganz weggelassen werden, wenn die Größe des Zählschrittes 1 ist. Ist in einem FOR-NEXT-Befehl keine STEP-Angabe gemacht worden, wird der Schleifendurchlauf automatisch in Einer-Schritten vollzogen.

Verschachtelte FOR-NEXT-Schleifen

Die Struktur des FOR-NEXT-Befehls wird als Programmschleife bezeichnet, da das Programm während der Ausführung dieser Anweisung zwischen FOR und NEXT und wieder zurück zu FOR hin und her pendelt. Da solche Schleifen sehr praktisch sind, werden Sie in beinahe allen Programmen angewendet. Sehr oft wird auch mit verschachtelten FOR-NEXT-Schleifen gearbeitet. Das bedeutet, daß die äußere, größte Schleife mehrere kleinere Schleifen in sich aufnimmt. Diese einzelnen FOR-NEXT-Befehle beinhalten in der Regel eine Vielzahl von Anweisungen. Abbildung 3-11 veranschaulicht Ihnen das Prinzip einfach verschachtelter Programmschleifen.
In Abbildung 3-12 sehen Sie sieben ineinander verschachtelte Schleifen. Die jeweils zwischen FOR und NEXT stehenden Anweisungen wurden wegen der besseren Übersichtlichkeit ausgelassen.
In diesem Beispiel hat die äußerste Schleife den Index N. Sie enthält drei weitere Schleifen mit den Variablen X, Y und Z. Die FOR-NEXT-Schleife mit dem Index X beinhaltet wiederum zwei zusätzliche Schleifen mit den Indizes A und B. In der Y-Schleife befindet sich noch eine weitere mit dem Index P.
Auch wenn es Ihnen im Moment noch nicht so erscheinen mag, sind Programmschleifen leicht zu benutzen und einfach nachzuvollziehen. Es gilt jedoch dabei auf etwas zu achten, das am Anfang häufig falsch gemacht wird: die innere Schleife muß vollständig innerhalb der äußeren Schleife liegen. Das heißt, daß das Ende der äußeren Schleife (NEXT . .) im Programm nicht vor dem NEXT der inneren Schleife stehen darf. In Abbildung 3-13 sehen Sie eine fehlerhafte Verschachtelung.

```
10 DIM A(99)
20 FOR N=0 TO 99 STEP 1
30 A(N)=N
40 REM DRUCKE ALLE WERTE VON A()
50 FOR J=0 TO N
60 PRINT A(J)
70 NEXT J
80 NEXT N
90 END
```

Abbildung 3-11: Einfache Verschachtelung

```
50 FOR N=1 TO 10
60 FOR X=25 TO 347 STEP 3
   .
   .
   .
100 FOR A=9 TO 0 STEP -1
   .
   .
   .
140 NEXT A
200 FOR B=25 TO 100 STEP 5
   .
   .
   .
280 NEXT B
300 NEXT X
   .
   .
   .
500 FOR Y=1 TO 20 STEP 2
   .
   .
   .
600 FOR P=10 TO 20
   .
   .
   .
650 NEXT P
700 NEXT Y
   .
   .
   .
1000 FOR Z=1 TO 10
   .
   .
   .
1090 NEXT Z
   .
   .
   .
1200 NEXT N
   .
   .
   .
```

Abbildung 3-12: Mehrfach verschachtelte FOR-NEXT-Schleifen

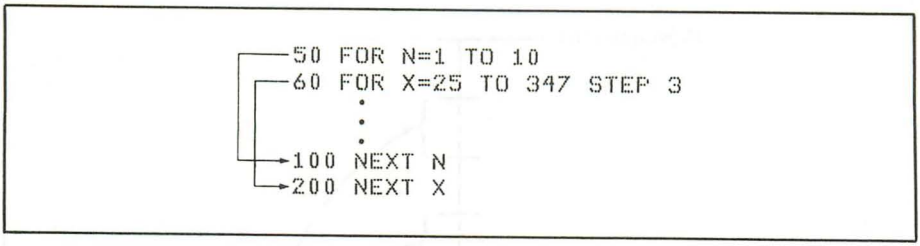


Abbildung 3-13: FALSCH verschachtelte FOR-NEXT-Schleifen

Auf diese Weise weiß der Computer immer, wo er mit der Programmausführung fortfahren soll, wenn er auf das dazugehörige NEXT stößt. Nachdem die FOR-NEXT-Schleife abgearbeitet ist, wird diese Notiz wieder aus dem internen Speicher gelöscht. Jede Programmschleife muß aus einer FOR- und einer NEXT-Anweisung bestehen. Wenn der Computer das zu einem FOR gehörende NEXT im Programm nicht findet, gibt er eine Fehlermeldung aus.

Unterprogramme

Sobald Sie anfangen, längere Programme zu schreiben, werden Sie feststellen, daß einzelne Programmabschnitte immer wieder verwendet werden. Nehmen Sie zum Beispiel an, daß den Elementen des Datenfeldes A an verschiedenen Stellen des Programms dieselben Werte zugewiesen werden sollen. Nun würden Sie wahrscheinlich die drei Anweisungen der FOR-NEXT-Schleife, die wir vorher besprochen haben, dementsprechend oft wiederholen. Solange man das nur mit drei Befehlen erledigen kann, ist dagegen nichts einzuwenden.
Nehmen wir aber an, mit der Definition der Variablen wären 10 weitere Anweisungen verbunden. Inzwischen sind es nun schon 13 Befehle geworden, die man für jede Wertzuweisung neu eintippen müßte. Das wäre nicht nur umständlich und zeitraubend, sondern es würde auch wichtigen Speicherplatz vergeuden (Abbildung 3-14).
Als Lösung bietet sich an, den Programmabschnitt, der öfter benötigt wird, herauszunehmen und von den verschiedenen Punkten des Programms zu ihm zu verzweigen. Einen solchen Programmabschnitt bezeichnet man als Unterprogramm.
Nun ergibt sich aber noch ein Problem. Vom Hauptprogramm zum Unterprogramm zu verzweigen, bereitet keine Schwierigkeiten, da ja auch das Unterprogramm mit Zeilennummern gekennzeichnet ist. Bis hierher könnte man also auch mit GOTO arbeiten. Was geschieht aber, wenn das Unterprogramm ausgeführt ist? (Abbildung 3-15) Haben mehrere GOTO-Anweisungen zur Unterroutine geführt, weiß der Computer nicht, zu welcher Stelle des Hauptprogramms er zurückverzweigen soll. Deshalb gibt es für diese Anwendungen den Befehl GOSUB.

GOSUB und RETURN

Der Befehl GOSUB führt die gleiche Programmverzweigung durch wie GOTO, merkt sich aber noch zusätzlich, zu welcher Zeile er zurückkehren muß (Abbildung 3-16).

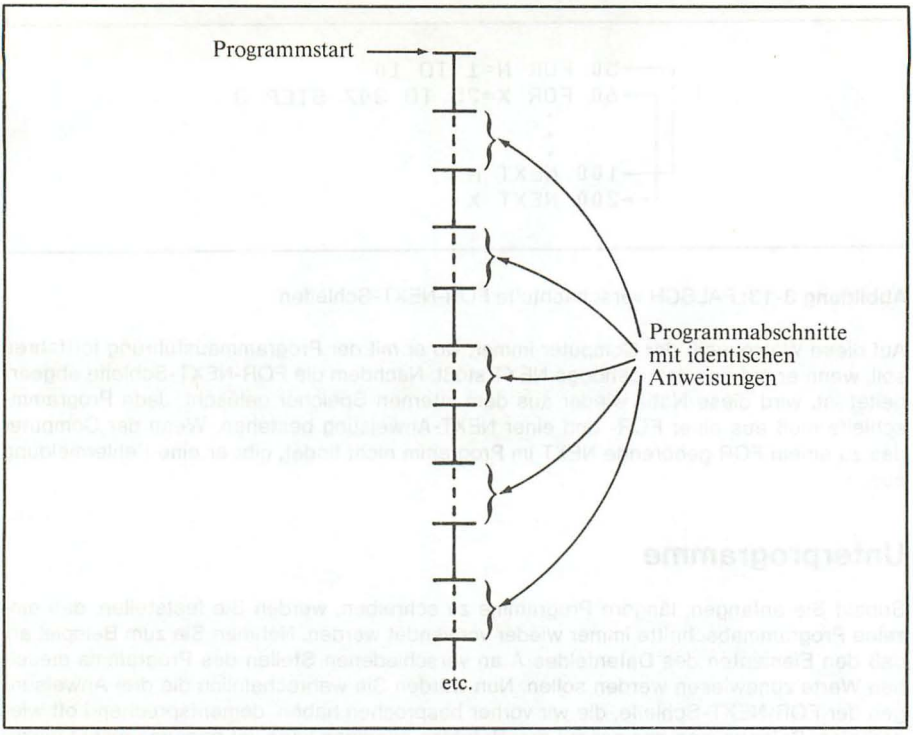


Abbildung 3-14: Der Speicherplatz wird mit den identischen Anweisungen vergeudet

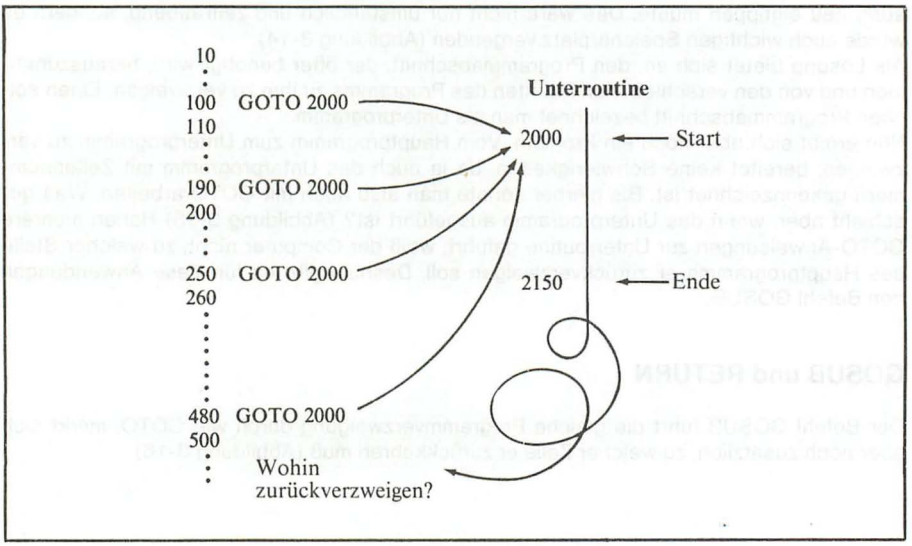


Abbildung 3-15: Verzweigen zu einer Unterroutine mit GOTO

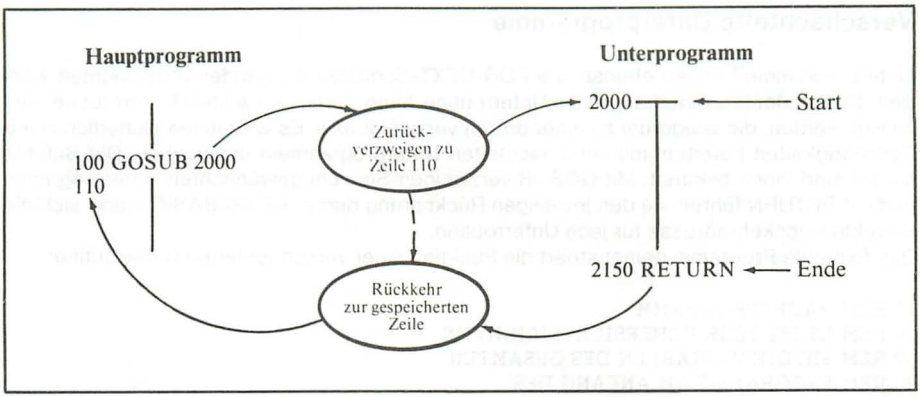


Abbildung 3-16: Verzweigung zu einem Unterprogramm mit GOSUB

Ein Unterprogramm wird mit einer RETURN-Anweisung abgeschlossen. Das veranlaßt den Computer, nach der Ausführung der Unterroutine zu dem Befehl zurückzuverzweigen, der GOSUB folgt. Wenn GOSUB die letzte Anweisung einer Programmzeile bildet, dann bewirkt RETURN, daß die Programmausführung beim ersten Befehl der folgenden Zeile wieder aufgenommen wird.
Im folgenden Beispiel sehen Sie die Initialisierung der Elemente des Datenfeldes A einmal auf eine andere Weise:

```
10 REM HAUPTPROGRAMM
20 REM ES IST SEHR UEBERSICHTLICH,WENN
30 REM SIE DIE VARIABLEN DES GESAMTEN
40 REM PROGRAMMS AM ANFANG DES
50 REM HAUPTPROGRAMMS DIMENSIONIEREN
60 DIM A(99)
70 GOSUB 2000
90 PRINT "ZURUECK IM HAUPTPROGRAMM"
100 END
2000 REM UNTERPROGRAMM
2010 FOR N=0 TO 99
2020 A(N)=N
2030 PRINT A(N)
2040 NEXT N
2050 RETURN
```

POP

In manchen Fällen erfordert es die Programmlogik, nach oder während der Ausführung der Unterroutine nicht zu der in RETURN gespeicherten Adresse zurückzukehren. Wenn sie in solcher Situation nur den Befehl GOTO eingeben, kann es zu Schwierigkeiten kommen, weil BASIC noch immer die Rückkehradresse von RETURN gespeichert hat. Für diese Fälle gibt es den Befehl POP. Er bewirkt, daß die in RETURN gespeicherte Zeilennummer gelöscht wird. Nachdem Sie diesen Befehl eingegeben haben, bestimmen Sie mit GOTO, wohin das Programm anschließend verzweigen soll.
Verwenden Sie den POP-Befehl jedoch nicht so häufig in Ihren Programmen, da diese ansonsten zu unübersichtlich werden.

Verschachtelte Unterprogramme

Unterprogramme können ebenso wie FOR-NEXT-Schleifen ineinander verschachtelt werden. Das bedeutet, innerhalb einer Unterroutine kann zu einer zweiten Unterroutine verzweigt werden, die wiederum zu einer dritten verzweigt usw. Es wird Ihnen sicherlich keine Schwierigkeiten bereiten, mit verschachtelten Unterprogrammen umzugehen. Die Befehle hierfür sind Ihnen bekannt: Mit GOSUB verzweigen Sie zum gewünschten Unterprogramm und mit RETURN führen sie den jeweiligen Rücksprung durch. ATARI-BASIC merkt sich die korrekte Rückkehradresse für jede Unterroutine.
Das folgende Programm demonstriert die Funktion einer verschachtelten Unterroutine:

```
10 REM HAUPTPROGRAMM
20 REM ES IST SEHR UEBERSICHTLICH,WENN
30 REM SIE DIE VARIABLEN DES GESAMTEN
40 REM PROGRAMMS AM ANFANG DES
50 REM HAUPTPROGRAMMS DIMENSIONIEREN
60 DIM A(99)
70 GOSUB 2000
90 PRINT "ZURUECK IM HAUPTPROGRAMM"
100 END
2000 REM UNTERPROGRAMM
2010 FOR N=0 TO 99
2020 A(N)=N
2030 GOSUB 3000
2040 NEXT N
2050 RETURN
3000 REM START D:VERSCHACHT.UNTERPR.
3010 PRINT A(N)
3020 RETURN
```

In diesem Beispiel wurde die Anweisung PRINT A(N) aus der ersten Unterroutine herausgenommen und in die zweite geschrieben. Wie Sie sehen, hat sich am Programmablauf nichts geändert.
Bitte beachten Sie noch etwas: In ATARI-BASIC kann ein Unterprogramm zwar andere aufrufen, nicht aber zu sich selbst verzweigen.
Die Zeilennummern nach GOSUB können Sie auch in Form eines arithmetischen Ausdrucks angeben:

```
100 GOSUB A*500+2000
110 REM
```

Der ATARI-Computer berechnet zuerst den Ausdruck in Zeile 100. Das Resultat ergibt die Anfangszeilennummer der Unterroutine, zu der verzweigt wird.

ON/GOSUB

Wie Sie gesehen haben, sind sich die Befehle GOTO und GOSUB sehr ähnlich. Deshalb ist es auch nicht verwunderlich, daß es einen ON/GOSUB-Befehl gibt, der einen "bedingten" Sprung veranlaßt und ON/GOTO in vieler Hinsicht gleich ist.
Zuerst berechnet der ATARI-Computer den Ausdruck oder den Variablenwert, der zwischen ON und GOSUB steht. Ist das Resultat gleich 0 oder aber größer als die Anzahl der hinter GOSUB aufgeführten Zeilennummern, wird keine Verzweigung durchgeführt und mit dem ON/GOSUB folgenden Befehl fortgefahren. Ergibt der Ausdruck einen negativen oder einen Wert größer als 255, führt dies zu einer Fehlermeldung.

Anschließend sehen Sie ein Beispiel der ON/GOSUB-Anweisung.

```
100 ON A GOSUB 1000,500,5000,2300
110 REM
```

Wenn A = 1, dann erfolgt der Sprung zu dem Unterprogramm, das mit der Zeilennummer 1000 startet. Ist A jedoch gleich 2, dann verzweigt das Programm zu der Unterroutine in Zeile 500 usw. Sollte der Zahlenwert von A anders lauten als 1, 2, 3 oder 4, wird die Programmausführung in Zeile 110 fortgeführt und nicht zu einer Unterroutine verzweigt.
Zur Verschachtelung von Unterprogrammen kann der ON/GOSUB-Befehl genauso angewendet werden wie GOSUB.

Bedingte Programmausführung

ON/GOTO und ON/GOSUB sind beides Befehle, die die Programmausführung nur dann fortsetzen, wenn eine entsprechende Bedingung erfüllt wurde. Das heißt, daß der korrekte Programmablauf vom Wert bzw. von der Bedingung verschiedener Variabler abhängt.

IF/THEN

IF/THEN ist eine weitere Anweisung, die gestellte Bedingungen prüft und danach verfährt. Ihre allgemeine Struktur sieht so aus:

IF-Ausdruck THEN-Befehl

Wenn die im Ausdruck gestellte Bedingung wahr ist, dann wird der nach THEN stehende Befehl ausgeführt. In der IF-THEN-Anweisung können Sie numerische, Boolesche und Relations-Ausdrücke verwenden. Dadurch ist Ihr BASIC-Programm auch in der Lage, Entscheidungen zu treffen. Nachfolgend sehen Sie drei einfache Beispiele für den Gebrauch von IF-THEN:

```
10 IF A=B+5 THEN PRINT MSG$
40 IF CC$="M" THEN IN=0
50 IF Q<14 AND M<M1 THEN GOTO 66
```

Die Programmzeile 10 sagt aus, dem PRINT-Befehl nachzukommen, wenn der Wert von A um fünf größer ist als der von B. Andernfalls wird die PRINT-Anweisung nicht ausgeführt. In Zeile 40 wird der numerischen Variablen IN der Wert 0 zugewiesen, wenn die Stringvariable CC$ den Buchstaben M beinhaltet. Zeile 50 stellt folgende Bedingung: Wenn Q kleiner ist als 14 und M ist kleiner als M1, dann soll mit der Programmausführung in Zeile 66 fortgefahren werden. Hierbei müssen beide Bedingungen erfüllt sein, ansonsten wird das Programm in der nächsten Zeile fortgesetzt.
Falls Sie Schwierigkeiten bei der Analyse der Ausdrücke zwischen IF und THEN haben, dann blättern Sie einfach zurück und lesen nochmals die Abschnitte durch, die sich mit den arithmetischen Ausdrücken beschäftigen.
In einer Programmzeile können dem IF-THEN-Befehl noch weitere Anweisungen folgen. ATARI-BASIC führt diese Anweisungen jedoch nur aus, wenn die zuvor gestellte Bedingung erfüllt ist. Wurde die Bedingung jedoch nicht erfüllt, dann wird das Programm mit dem ersten Befehl der nächsten Programmzeile fortgeführt. Das veranschaulichen die folgenden Zeilen:

```
10 IF V>100 THEN PRINT "22.FEBRUAR";GOSUB 2000
20 T=T+V:PRINT T
```

Diese Bedingung lautet: Wenn die Variable V größer ist als 100, dann drucke die Meldung 22.FEBRUAR und rufe die Unterroutine in Zeile 2000 auf. Wenn die gestellte Bedingung jedoch nicht erfüllt ist, wird anstatt dessen die Zeile 20 ausgeführt.
ATARI-BASIC bietet Ihnen noch eine Erleichterung an. Wenn Sie nach THEN eine Zeilennummer als Sprungziel eingeben wollen, so können Sie das Wort GOTO einfach weglassen und nur die Zeile angeben. Die beiden untenstehenden Anweisungen veranlassen dasselbe:

```
10 IF MM$=DD$ THEN GOTO 100
```

```
10 IF MM$=DD$ THEN 100
```

Eingabe- und Ausgabe-Anweisungen

Es gibt eine Vielzahl von BASIC-Befehlen, die den Datenfluß zwischen dem Computer und den angeschlossenen Peripheriegeräten steuern. Diese Befehle werden zusammenfassend als Ein/Ausgabe-Anweisungen bezeichnet. Die einfachsten dieser Anweisungen, die Sie in den nächsten Paragraphen kennenlernen werden, kontrollieren die Dateneingabe über die Tastatur sowie die Datenausgabe auf dem Bildschirm. Es gibt aber auch kompliziertere Ein/Ausgabe-Anweisungen, die die Datenübertragung zwischen Computer und Peripheriegeräten wie Kassettenrekorder, Diskettenlaufwerk und Drucker steuern. Diese Befehle werden in den folgenden Kapiteln ausführlich erklärt.
Wir wollen uns zuerst mit einem Befehl beschäftigen, der die Daten auf dem Bildschirm ausgibt. Sie haben ihn schon häufig benutzt; es ist der PRINT-Befehl.

PRINT

Sicherlich haben Sie sich schon gefragt, warum man anstatt des Wortes PRINT (zu deutsch = drucken) nicht einen anderen, passenderen Ausdruck, wie zum Beispiel "darstellen" oder "schreiben", gewählt hat. Nun, die Antwort ist einfach. Anfang der 60er Jahre, als die Programmiersprache BASIC entwickelt wurde, waren Bildschirme noch sehr teuer und somit für Computer der mittleren und unteren Preiskategorie auch meistens nicht verfügbar.
Eine typische Computerstation bestand damals aus einem Tastenfeld und einem Drucker, auf dem die Texte und Zahlen ausgedruckt wurden, während sie heutzutage auf dem Bildschirm dargestellt werden. Soweit zum Ursprung des Wortes PRINT.
Die folgende Anweisung bewirkt, daß das Wort TEXT dargestellt wird:

```
10 PRINT "TEXT"
```

Zur Darstellung von Zahlen schreiben Sie einfach die gewünschte Zahl oder den Variablennamen hinter PRINT:

```
10 A=10
20 PRINT 5,A
```

Durch die Anweisung in Zeile 20 werden die Zahlen 5 und 10 in derselben Bildschirmzeile ausgegeben.
In einem PRINT-Befehl kann man Texte und Zahlen in beliebiger Reihenfolge unterbringen. Die einzelnen Angaben werden durch Kommas oder Semikolons voneinander getrennt. Die folgende Programmzeile stellt die Worte EINS bis FUENF und dazwischen die jeweiligen Zahlen dar.

```
10 PRINT "EINS",1,"ZWEI",2,"DREI",3,"VIER",4,"FUENF",5
20 END
```

Trennen Sie die einzelnen Angaben durch Kommas, wie in dem obigen Beispiel, dann fügt der ATARI-Computer bei der Darstellung automatisch eine bestimmte Anzahl von Leerzeichen zwischen den Daten ein. Tippen Sie nun die beiden Programmzeilen ein. Anschließend ersetzen Sie die Kommas durch Semikolons und schreiben Sie RUN.

```
10 PRINT "EINS";1;"ZWEI";2;"DREI";3;"VIER";4;"FUENF";5
20 END
```

Sie stellen fest, daß die Daten nun ohne Zwischenleerzeichen dargestellt werden. Nachdem die PRINT-Anweisung ausgeführt wurde, springt der Cursor automatisch an den Anfang der nächsten Bildschirmzeile (RETURN). Dieser Zeilensprung läßt sich unterdrücken, indem man nach der letzten Angabe der PRINT-Anweisung ein Komma oder ein Semikolon eingibt. Diese beiden Zeichen veranlassen, daß der Cursor an die Stelle rückt, an der – falls vorhanden – der nächste PRINT-Befehl dargestellt wird.
Probieren Sie die folgenden Programmbeispiele selbst aus, um sich die Funktion dieser beiden Zeichen einzuprägen:

```
10 PRINT "EINS",1
20 PRINT "ZWEI",2
30 END
```

Die Ausgabe erstreckt sich über zwei Bildschirmzeilen. Nun setzen Sie ein Komma an das Ende der Zeile 10 und starten das Programm erneut. Wie Sie sehen, werden die Daten jetzt in nur einer Zeile dargestellt.
Ersetzen Sie nun das Komma am Ende der Zeile 10 auch durch ein Semikolon, und lassen Sie das Programm nochmals ablaufen. Die Ausgabedaten erscheinen wiederum in einer Bildschirmzeile, jedoch wurden die Leerzeichen zwischen der Zahl 1 und dem Wort ZWEI nicht dargestellt. Durch die Umwandlung der übrigen Kommas in Semikolons werden auch noch die restlichen Zwischenräume unterdrückt.
In den bisher demonstrierten Beispielen haben wir Zahlen, die dargestellt werden sollten, immer direkt in die PRINT-Anweisung getippt. Mit diesem Befehl können Sie aber ebenso die Werte definierter Variablen ausgeben lassen. In dem folgenden Programm werden zuerst den Elementen des Datenfeldes A Werte zugewiesen und diese dann anschließend, in der gleichen Form wie in einem vorhergehenden Beispiel, dargestellt. Geben Sie diese Zeilen ein:

```
5 DIM A(5)
10 FOR N=1 TO 5
20 A(N)=N
30 NEXT N
40 PRINT "EINS";A(1);"ZWEI";A(2);"DREI";A(3);"VIER";A(4);"FUENF";A(5)
50 END
```

Eine weitere Variante des Beispielprogramms wäre, die Worte EINS bis FUENF mittels READ/DATA ausgeben zu lassen:

```
10 DIM N$(5)
20 DATA EINS,ZWEI,DREI,VIER,FUENF
30 FOR N=1 TO 5
40 READ N$
50 PRINT N$;N;
60 NEXT N
```

```
69 REM CURSOR Z. NAECHSTEN ZEILENANFANG
70 PRINT
80 END
```

Die PRINT-Anweisung in Zeile 70 bewirkt, daß der Cursor an den Anfang der nächsten Zeile springt.

INPUT

Trifft der Computer bei der Ausführung eines BASIC-Programms auf die Anweisung INPUT, dann stoppt er die Programmausführung solange, bis die Daten, die er benötigt, eingegeben worden sind.
In seiner einfachsten Form besteht dieser Befehl aus dem Wort INPUT und einem Variablennamen. Der Computer ordnet die über die Tastatur eingegebenen Daten der INPUT-Variablen zu. Der Variablentyp nach dem Wort INPUT bestimmt die Eingabeart, die der Computer erwartet. Das bedeutet: Stringvariablen darf nur Text zugewiesen werden und numerischen Variablen nur Ziffern. Geben Sie nun das folgende Kurzprogramm ein und starten Sie es. Versuchen Sie auch einmal, Buchstaben einzutippen, und beobachten Sie, was passiert:

```
10 INPUT A
20 PRINT A
25 REM WENN EINGABE=0 DANN PROGR.ENDE
30 IF A=0 THEN END
40 GOTO 10
```

Wie Sie gesehen haben, stellt der Computer bei der Ausführung des PRINT-Befehls ein Fragezeichen dar und wartet auf Ihre Eingabe. Wenn Sie diese durch Drücken der RETURN-Taste abgeschlossen haben, wird die Anweisung in Zeile 20 ausgeführt. Diese veranlaßt, daß die zuvor eingegebene Zahl auf dem Bildschirm dargestellt wird. Somit erscheint diese Ziffer zweimal während des Programmablaufes.
Man kann den INPUT-Befehl auch so formulieren, daß er die Eingabe mehrerer Texte oder Zahlen verarbeitet. In diesem Fall werden die einzelnen Variablen durch Kommas getrennt. INPUT erwartet, daß für jede Variable ein separater Wert eingetippt wird und die Eingaben mit RETURN abgeschlossen werden.
Im folgenden Beispiel werden zwei numerische Werte verlangt, die anschließend auf dem Bildschirm ausgegeben werden:

```
20 INPUT A,B
30 PRINT A,B
35 REM WENN EINGABE=0 DANN PROGR.ENDE
40 IF A=0 OR B=0 THEN END
50 GOTO 20
```

Lassen Sie das Programm ablaufen und tippen Sie eine Zahl, anschließend ein Komma und dann wieder eine Zahl. Nun drücken Sie RETURN. Jetzt versuchen Sie noch etwas anderes. Schreiben Sie nur eine Zahl und drücken Sie gleich RETURN. Wie Sie sehen, erinnert Sie der ATARI-Computer daran, auch den zweiten Wert einzugeben. Nachdem Sie das getan haben, drücken Sie wieder RETURN.
Wenn ein INPUT-Befehl mit mehreren numerischen Variablen ausgeführt wird, dann haben Sie die Wahl, ob Sie alle Ziffern, durch Kommas getrennt, in eine Reihe schreiben oder jede einzeln in eine separate Zeile tippen.

Bei Stringvariablen sind noch einige Besonderheiten zu berücksichtigen. Geben Sie folgendes Programm ein:

```
10 DIM A$(9)
20 INPUT A$
29 REM WENN NICHTS EINGEGEBEN WIRD, PROGRAMM BEENDEN
30 IF A$=" " THEN END
40 PRINT A$
50 GOTO 20
```

Versuchen Sie nun, A$ mehr als neun Zeichen zuzuweisen, und achten Sie darauf, was passiert. Da A$ 9 als höchsten Index hat, ignoriert der Computer alle zusätzlichen Zeichen.
Bei der Texteingabe für eine Stringvariable ist darauf zu achten, daß die einzelnen Daten jeweils in eine separate Zeile geschrieben werden. Anders als bei INPUT mit numerischen Variablen würde ATARI-BASIC das zur Trennung der Daten erforderliche Komma den Strings zuordnen. Zum Beweis dafür lassen Sie das obige Programm nochmals ablaufen und geben PAUL,NINA ein.
Das folgende Beispiel zeigt Stringvariablen und numerische Variablen zusammen in einem INPUT-Befehl. Geben Sie das Programm ein und experimentieren Sie damit. Versuchen Sie zum Beispiel, alle Werte in eine Zeile zu schreiben. Beobachten Sie aufmerksam, was anschließend passiert.

```
10 DIM A$(10),B$(10)
20 INPUT A$,A,B$,B
30 PRINT A$,A,B$,B
35 REM WENN NICHTS EINGEGEBEN WIRD, PROGRAMM BEENDEN
40 IF A$=" " THEN END
50 GOTO 20
```

Editieren der Dateneingaben

Wenn Sie die Antwort auf eine INPUT-Anweisung eintippen, können Sie alle Editier-Möglichkeiten, die der ATARI-Computer bietet, ausschöpfen: die Pfeil-Tasten, die INSERT- und DELETE-Tasten, die TAB- und die BACK-S-Taste. Die Tastenfunktionen sind die gleichen wie beim Schreiben von Programmzeilen. Vergessen Sie jedoch nicht, unmittelbar nach der korrekten Eingabe RETURN zu drücken.

Eingabeanweisungen für den Anwender

In manchen Fällen ist es für den Anwender schwer zu erraten, welche Daten er eingeben soll. Deshalb ist es sehr hilfreich, wenn vor dem INPUT-Fragezeichen erklärender Text dargestellt wird. Jemand, der nichts über Computer und das Programmieren weiß, wird sehr schnell frustriert aufgeben, wenn er bei Dateneingaben unverständliche Fehlermeldungen erhält, die letzten Endes vielleicht nur durch ein zusätzliches Komma ausgelöst wurden. Sie können mit großem Zeitaufwand "idiotensichere" Programme schreiben, die jeden Fehler, den der Anwender bei der Dateneingabe machen könnte, abdeckt. Dazu könnte man auch noch Tips zur korrekten Eingabe erteilen.
Es gibt dafür jedoch auch eine einfachere Lösung, indem Sie dem Anwender mit wenigen Worten erklären, welche Daten er eingeben darf. Diese bringen Sie in einem PRINT-Befehl unter, der unmittelbar vor dem INPUT steht. Das INPUT-Fragezeichen wird direkt nach dem erklärenden Text dargestellt, wenn Sie ans Ende der PRINT-Anweisung ein Semikolon setzen.

```
9 REM MULTIPLIKATIONS TEST
10 FOR N=1 TO 0
20 PRINT "WIEVIEL SIND ";N;"*9";
30 INPUT ANT
39 REM BEI FALSCHER ANTWORT WIRD DIE FRAGE WIEDERHOLT
40 IF ANT<>N*9 THEN GOTO 20
49 REM ANSONSTEN ZUR NAECHSTEN AUFGABE
50 PRINT "SEHR GUT!"
60 NEXT N
70 END
```

Anhalten und Fortsetzen der Programmausführung

Wenn Sie eine Programmausführung unterbrechen wollen, dann drücken Sie die Taste BREAK. Der Computer gibt folgende Meldung aus:

```
STOPPED AT LINE 1200
```

Anstatt 1200 wird die Zeilennummer dargestellt, an der das Programm gestoppt wurde. BREAK unterbricht die Programmausführung in dem Augenblick, in dem die Taste gedrückt wurde.

Der Befehl CONT bewirkt dagegen, daß das unterbrochene Programm fortgeführt wird. Allerdings wird die Programmausführung erst in der folgenden BASIC-Zeile wieder aufgenommen. Wurde das Programm in der Mitte einer Zeile gestoppt, so werden bei der Weiterführung die restlichen Befehle dieser Zeile nicht mehr ausgeführt. Sehen Sie dazu das untenstehende Beispiel (Abbildung 3-17). Wenn Sie BREAK betätigen, während der Computer den INPUT-Befehl in Zeile 50 durchführt, springt das Programm, nachdem Sie wieder CONT getippt haben, gleich zu Zeile 60. Dadurch wird die Zeile 50 nicht korrekt bearbeitet, was natürlich im weiterführenden Programm Schwierigkeiten verursacht. Probieren Sie das nun selbst aus.

```
9 REM ANALYSE DER MONATLICHEN AUSGABEN
10 DIM AUSGB$(12),FESTKOSTEN(4)
20 DATA MIETE,TELEFON,GAS,STROM,LEBENSMITTEL
29 REM HOEHE DER ASUGABEN
30 FOR N=0 TO 4
40 READ AUSGB$:PRINT AUSGB$
50 INPUT X:FESTKOSTEN(N)=X
60 NEXT X
69 REM EINKOMMENSANGABE
70 PRINT:PRINT "EINKOMMEN";
80 INPUT EINKOMMEN
89 REM VERGLEICH ZWISCHEN EINK.&AUSGB.
90 PRINT:PRINT:PRINT "ANALYSE---"
100 PRINT
110 RESTORE
120 FOR N=0 TO 4
130 READ AUSGB$
139 REM BERECHNE DEN PROZENTSATZ
140 PRINT AUSGB$;" BETRAEGT ";FESTKOSTEN(N)/EINKOMMEN*100;"
% DES EINKOMMENS"
150 NEXT N
160 END
```

Abbildung 3-17: Programmausdruck

Die Taste SYSTEM RESET

Man kann das Programm auch unterbrechen, indem man die Taste SYSTEM RESET drückt. Allerdings bewirkt das, daß der Bildschirm gleichzeitig gelöscht wird und der Computer zur direkten Betriebsart zurückkehrt. Sie können mit CONT versuchen, die Programmausführung fortzusetzen. Diese setzt dann aber auch erst wieder in der folgenden Zeile ein. Haben Sie ein Programm mit SYSTEM RESET unterbrochen, ist die Chance der korrekten Programmfortführung um so kleiner, je umfangreicher das Programm ist.

END

Den END-Befehl, mit dem eine Programmausführung beendet wird, haben wir zuvor schon einmal erwähnt. Der Computer stellt nach diesem Befehl die Meldung READY dar und kehrt zur direkten Betriebsart zurück.
Mit der Anweisung CONT kann man veranlassen, daß die Programmausführung fortgesetzt wird. Dies geschieht jedoch auch erst wieder in der folgenden Zeile. Die Befehle, die nach END in einer Zeile stehen, werden also nach der Eingabe von CONT nicht ausgeführt. Fügen Sie die folgenden Programmzeilen in das Beispiel aus der Abbildung 3-17 ein:

```
65 END:? "HALLO"
```

Lassen Sie das Programm ablaufen und tippen Sie CONT, sobald es stoppt. Nach dem Drücken der RETURN-Taste wird die Ausführung in Zeile 70 fortgesetzt. Das bedeutet, daß der PRINT-Befehl in Zeile 65 nie bearbeitet wird.

STOP

ATARI-BASIC kennt noch einen weiteren Befehl, der die Programmausführung unterbricht: den Befehl STOP. Bei dieser Anweisung gibt der Computer folgende Meldung aus:

```
STOPPED AT LINE 1200
```

Anstatt 1200 wird die Zeilennummer dargestellt, an der das Programm gestoppt wurde. Der Befehl bewirkt außerdem eine Rückkehr zur direkten Betriebsart.
Auch hier kann die Programmausführung mit CONT fortgesetzt werden. Diese setzt, ebenso wie bei den vorhergehenden Befehlen, erst wieder in der nächsten Zeile ein. Fügen Sie nun diese Programmzeile in das letzte Beispiel (Abbildung 3-17) ein:

```
65 STOP:? "WIE GEHT'S?"
```

Starten Sie das Programm erneut und tippen Sie CONT, sobald es stoppt. Da die Ausführung in Zeile 70 fortgeführt wird, kann die Anweisung in Zeile 65 nicht bearbeitet werden.

Funktionen

Ein weiteres BASIC-Element bilden die Funktionen. Diese erscheinen oftmals wie Variablen, haben aber die Eigenschaften von BASIC-Befehlen. In den folgenen Abschnitte werden die Funktionen im einzelnen erklärt. Kapitel 11 enthält eine komplette alphabetische Liste aller BASIC-Funktionen.
Sehen Sie dazu folgendes Beispiel:

```
10 A=SQR(B)
```

Der Variablen A wird die Quadratwurzel der Variablen B zugewiesen. Das Schlüsselwort SQR spezifiziert die Wurzelbildung.
Jetzt eine String-Funktion:

```
20 L=LEN(D$)
```

Der numerischen Variablen L wird die Länge der Stringvariablen D$ zugeordnet.
Alle Funktionen, bis auf eine, haben das gleiche Format (Abbildung 3-18). Die einzelnen Funktionen werden durch bestimmte Schlüsselworte, wie SQR für Quadratwurzel, spezifiert. Danach steht ein in Klammern gesetztes Argument (die Ausnahme bildet hierbei USR, dem mehrere Argumente folgen können).
Eine Funktion berechnet den Wert des Argumentes. Deshalb kann dieser gesamte Ausdruck, genau wie Variablen oder Konstante, Teil einer BASIC-Anweisung sein. Einige Funktionen können mit numerischen Werten, andere mit Strings verwendet werden. SQR zum Beispiel berechnet die Quadratwurzel eines numerischen Argumentes. LEN dagegen zählt die Zeichen einer Stringvariablen oder eines Textes. Variable oder Konstante können überall innerhalb einer BASIC-Anweisung durch eine Funktion ersetzt werden, ausgenommen links von einem Gleichheitszeichen. Mit anderen Worten ist A=SQR(B) richtig, SQR(B)=A dagegen falsch.
Funktionen, die Teil einer Anweisung sind, werden berechnet und in einen Wert umgewandelt, bevor der Rest der Anweisung ausgeführt wird. Argumente können Konstante, Variable oder Ausdrücke sein. Deshalb muß der Computer wiederum, bevor er die Funktion ausführen kann, den Ausdruck berechnen. Das folgende Beispiel enthält mehrere Funktionen, die alle vor der eigentlichen Berechnung des Ausdrucks bearbeitet werden müssen.

```
10 B=24.7*(SQR(C)+5)-SIN(0.2+D)
```

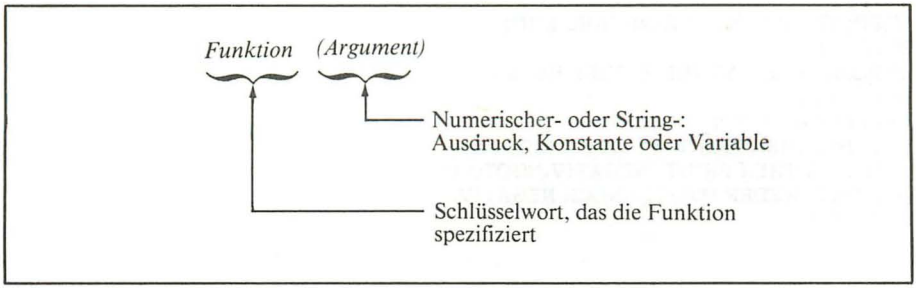


Abbildung 3-18: Format einer Funktion

Zuerst berechnet der ATARI-Computer die Funktionen SQR und SIN. Wir nehmen an, daß SQR(C)=6.72 und SIN(0.2+D) ist. Danach stellt sich uns der Ausdruck so dar: 24.7*(6.72+5)–0.625. Am Ende des Rechenvorgangs hat die Variable B den Wert 288.859.

Numerische Funktionen

Nachfolgend sehen Sie eine Liste der numerischen Funktionen, die Sie in ATARI-BASIC verwenden können:

SGN	stellt das Vorzeichen fest; ergibt +1 für ein positives Argument, –1 für ein negatives Argument und 0 wenn, das Argument gleich Null ist.
ABS	stellt den absoluten Wert eines Argumentes fest. Ein positives Argument bleibt unverändert, während ein negatives Argument in sein positives Äquivalent umgewandelt wird.
RND	erzeugt eine Zufallszahl zwischen 0 und 1.
INT	wandelt Bruchzahlen in ganze Zahlen um.
SQR	errechnet die Quadratwurzel.
EXP	errechnet die Zahl e(2.71828179) hoch dem Wert des Argumentes.
LOG	errechnet den natürlichen Logarithmus.
CLOG	errechnet den dekadischen Logarithmus.
SIN	errechnet den Sinus des Argumentes.
COS	errechnet den Cosinus des Argumentes.
ATN	errechnet den Arcustangens des Argumentes.

Setzen Sie Funktionen in Ihren Programmen überall dort ein, wo sie helfen, Ihre Arbeit zu erleichtern. Verwenden Sie jedoch keine, deren Bedeutung Sie noch nicht verstanden haben. Wenn Sie zum Beispiel mit der Trigometrie nicht vertraut sind, sollten Sie die Funktionen SIN, COS und ATN nicht benutzen. Geben Sie das folgende Programm ein, das herausfindet, ob die Zahlen, die Sie eingegeben haben, negativ oder positiv sind.

```
10 PRINT "GEBEN SIE EINE ZAHL EIN";
20 INPUT A
29 REM FINDE DAS VORZEICHEN HERAUS
30 B=SGN(A)
40 PRINT A;" IST ";
50 IF B=1 THEN PRINT "POSITIV.":GOTO 10
60 IF B=-1 THEN PRINT "NEGATIV.":GOTO 10
70 PRINT "WEDER POSITIV NOCH NEGATIV."
80 GOTO 10
```

DEG und RAD

Die drei trigometrischen Funktionen SIN, COS und ATN benutzen normalerweise zur Winkelberechnung das Bogenmaß. Mit dem Befehl DEG kann man die Winkelfunktion auf Grad umschalten. Dagegen wird mit dem Befehl RAD die Winkelfunktion wieder auf das Bogenmaß zurückgeschaltet. Dazu einige Beispiele:

```
DEG

READY
?SIN(90)
1

READY
RAD

READY
?SIN(1.571)
0.9999999833

READY
```

String-Funktionen

String-Funktionen ermöglichen Ihnen, String-Daten auf vielfältige Weise zu manipulieren. Anschließend sehen Sie eine Liste dieser Funktionen, die Sie in ATARI-BASIC benutzen können:

ADR	ermittelt die Lage des Strings im Speicher.
ASC	liefert den numerischen Wert des ATASCII-Codes für das erste Zeichen eines Strings.
LEN	stellt die Länge einer Stringvariablen fest.
STR$	wandelt einen numerischen Wert in eine Stringvariable um.
VAL	wandelt eine Stringvariable in einen numerischen Wert um.
CHR$	wandelt eine ATASCII-Codezahl in die entsprechende Stringvariable um.

Nun einige Beispiele für die Anwendung der String-Funktionen:

STR$(14)	verwandelt 14 in "14".
LEN("ABC")	berechnet die Länge des Strings. In diesem Fall 3.
VAL("14")	verwandelt "14" in 14.

System-Funktionen

Zum Schluß noch einige Funktionen, deren Anwendung in Kapitel 4 noch näher beschrieben wird.

PEEK	liest den Inhalt einer Speicherzelle.
FRE	errechnet die Anzahl der verfügbaren Speicherzellen.
USR	läßt den Computer ein Maschinenprogramm ausführen.
PADDLE	ermittelt die Stellung des Drehknopfes beim Drehregler.
PTRIG	stellt fest, ob der Druckknopf des Drehreglers gedrückt ist.
STICK	ermittelt die Stellung des Steuerknüppels.
STRIG	stellt fest, ob der Feuerknopf des Steuerknüppels gedrückt ist.

4

BASIC FÜR FORTGESCHRITTENE

Dieses Kapitel beschäftigt sich mit der detailierten Beschreibung der Programmiersprache BASIC. Es schließt an das Kapitel 3 an, wo Sie den Umgang mit kleinen Programmen erlernt haben. Nun werden Sie viele neue Befehle und aufwendigere Programmiertechniken kennenlernen.

Strings

Früher war es schwierig mit dem Computer zu kommunizieren, da er nur Zahleneingaben akzeptieren konnte. ATARI-BASIC ermöglicht Ihnen, auch mit Zeichen und Buchstaben zu arbeiten. Diese werden einem String zugewiesen. Deshalb ist es sehr wichtig, soviel wie möglich über Strings und deren Handhabung zu lernen.

Wie Strings gespeichert werden

Bevor Sie nun mit einem String arbeiten, sollten Sie wissen, wie die Zeichen im Computerspeicher abgelegt werden. Das Konzept ist einfach zu verstehen. Der Computer wandelt alle Zeichen intern in Zahlen um, da sein Speicher nur Zahlen aufnimmt. Der ATARI-Computer bedient sich bei dieser Umwandlung eines speziellen Codes. Abweichend zu den meisten anderen Computern, die den ASCII (Amerikanischer Fernschreibcode) verwenden, benutzt der ATARI-Computer den ATACSII-Code (ATARI-ASCII). Zum Beispiel ist der ATASCII-Code für A=65, für B=66, C=67 und so weiter. In Anhang D finden Sie eine komplette Liste des ATASCII-Codes und den dazugehörenden Zeichen.
Die ASC-Funktion wandelt das erste Zeichen eines Strings in seine ATASCII-Codezahl um. Sehen Sie dazu das folgende Programm:

```
10 DIM A$(1)
20 PRINT "GEBEN SIE EIN ZEICHEN EIN";
30 INPUT A$
40 PRINT "DER ATASCII CODE FUER ";A$;" IST:"
50 PRINT ASC(A$)
59 REM PROGRAMM STOPPEN M. BREAK TASTE
60 GOTO 20
RUN
GEBEN SIE EIN ZEICHEN EIN?A
DER ATASCII CODE FUER A IST:
65
GEBEN SIE EIN ZEICHEN EIN?8
DER ATASCII CODE FUER 8 IST:
56
GEBEN SIE EIN ZEICHEN EIN?
```

Funktionstasten

Haben Sie schon mal versucht, eine der Cursorsteuerungstasten, wie ← (CTRL-+), einer Stringvariablen zuzuweisen? Oder etwa diese Funktion in einem PRINT-Befehl unterzubringen? Nun wahrscheinlich hatten Sie damit kein Glück.
BASIC kennt aber Möglichkeiten, daß diese Sonderzeichen in einem String akzeptiert werden. Dazu drücken Sie zuerst die Taste ESC, anschließend die Tastenkombination, die das gewünschte Zeichen erzeugt. Sie sehen in der folgenden Tabelle zum Beispiel den Ausdruck ESC/CTRL-+, was bedeutet: Drücken Sie die Taste ESC, lassen diese los und drücken Sie danach die CTRL und die +Taste gleichzeitig. Wie Sie feststellen können, hat sich der Cursor nicht bewegt. Ohne das Drücken der ESC-Taste wäre er nach links gerückt. Anstattdessen wird ein Zeichen erzeugt, das den ATASCII-Code 30 hat. Erst bei der Ausführung dieses Zeichens bewegt sich der Cursor nach links. Tabelle 4-1 zeigt eine vollständige Liste der Funktionstasten mit deren Funktionen.
Die Zeichen, die durch die Funktionstasten auf dem Bildschirm dargestellt werden, sind nicht identisch mit deren eigentlichen Funktionen. Wenn Sie zum Beispiel die Tasten ESC/CTRL -+ drücken, erhalten Sie dieses Zeichen ←. Seine Funktion können Sie in der folgenden Zeile prüfen.

```
PRINT "ABC←←D"
ADC

READY
```

Sie haben ebenso die Möglichkeit, alle Funktionstasten, sofern Sie diese in einem String oder in einer PRINT-Anweisung verwenden, auch darstellen zu lassen. Um dies zu erreichen, leiten Sie jede Funktionstaste mit der Sonderfunktion ESC/ESC ein. Probieren Sie dazu folgendes Beispiel (E_L ist das Zeichen für ESC/ESC):

```
PRINT "E↓E↓"
↓↓

READY
```

ATARI-BASIC ermöglicht Ihnen, Zeichen zu produzieren, indem Sie direkt die jeweilige ATASCII-Code-Nummer eingeben. Die CHR$-Funktion übersetzt die Code -Nummer in das äquivalente Zeichen. Um das Zeichen "$" zu erzeugen, suchen Sie in Anhang D den ATASCII-Code heraus. Dann geben Sie die Funktion CHR$ wie folgt ein:

Tabelle 4-1: Funktionstasten-Verzeichnis

Tasten-kombination	Darge-stelltes Zeichen	ATASCII-Code	Funktion
ESC/ESC		27	Escape Code
ESC/BACK S		126	bewegt Cursor nach links, löscht Zeichen unter dem Cursor
ESC/TAB		127	Cursor springt nach rechts zur nächsten Tabulatorstelle
ESC/CTRL--		28	bewegt Cursor nach oben
ESC/CTRL-=		29	bewegt Cursor nach unten
ESC/CTRL-*		30	bewegt Cursor nach rechts
ESC/CTRL-+		31	bewegt Cursor nach links
ESC/CTRL-BACK S		254	löscht Zeichen
ESC/CTRL->		255	fügt Zeichen ein
ESC/CTRL-<		125	löscht Bildschirm
ESC/CTRL-TAB		158	löscht Tabulatorstelle
ESC/CTRL-2		253	erzeugt Summton
ESC/SHIFT-BACK S		156	löscht Zeile
ESC/SHIFT->		156	fügt Zeile ein
ESC/SHIFT-<		125	löscht Bildschirm
ESC/SHIFT-TAB		159	setzt Tabulatormarke

```
PRINT CHR$(36)
$

READY
```

Experimentieren Sie nun selber mit den ATASCII-Code-Nummern, die zwischen 0 und 255 liegen.
Wie Sie der folgenden Programmzeile entnehmen können, kann man die CHR$-Funktion auch zusammen mit normalen Texten in einem PRINT-Befehl benutzen.

```
?CHR$(34);"KOENIGIN VON ";CHR$123);
CHR$(34);CHR$(253)
"KOENIGIN VON ♠"

READY
```

Mit dieser Funktion haben Sie ferner die Möglichkeit, den Strings solche Zeichen zuzuweisen, die ansonsten falsch interpretiert werden würden, wie zum Beispiel die Anführungsstriche innnerhalb eines Strings.

Teilstrings

Man kann aus dem Inhalt eines Strings einen bestimmten Teil herausziehen. Um einen solchen Teilstring zu erhalten, gibt man zuerst den Namen der Stringvariablen an. Danach geben Sie die erste und die letzte Position der Zeichen an, die Sie benötigen. Diese Angabe muß eingeklammert sein. Nehmen wir an, die Stringvariable A$ hätte folgenden Inhalt: ABCDEF. Der Teilstring BCD wird durch A$(2,4) spezifiziert und enthält das zweite, dritte und vierte Zeichen von A$.
Die letzte Position einer Teilstring-Bestimmung muß nicht angegeben werden. Entfällt

diese Angabe, dann nimmt ATARI-BASIC an, daß Sie den ganzen verbleibenden Teil des Strings benötigen. Das ist alles, was rechts von der ersten spezifizierten Position steht. Bei der Angabe A$(2) werden also die Zeichen BCDEF ausgegeben.
Die Zeichenpositionen können auch in Form von numerischen Konstanten, Variablen oder Ausdrücken angegeben werden.
Ein Teilstring kann auch auf der linken Seite eines Zuordnungsbefehls stehen, wie im folgenden Programm:

```
10 DIM A$(20)
20 A$="VORNAME"
30 A$(4,7)="TEIL"
40 ?A$
RUN
VORTEIL

READY
```

Der ATARI-Computer gibt die Fehlermeldung 5 aus, wenn der Teilstring nicht korrekt spezifiziert wurde. Zum Beispiel kann man nicht schreiben A$(4,3) oder A$(7,1), da das letzte Zeichen eines Teilstrings nicht vor dem ersten angegeben werden darf. Außerdem erlaubt BASIC keine 0 als Positionsangabe.

Strings verbinden

Sie können Strings miteinander verbinden, um einen langen String zu formen (Abbildung 4-1). Durch dieses Aneinanderreihen, kann man Strings von beliebiger Länge schaffen.

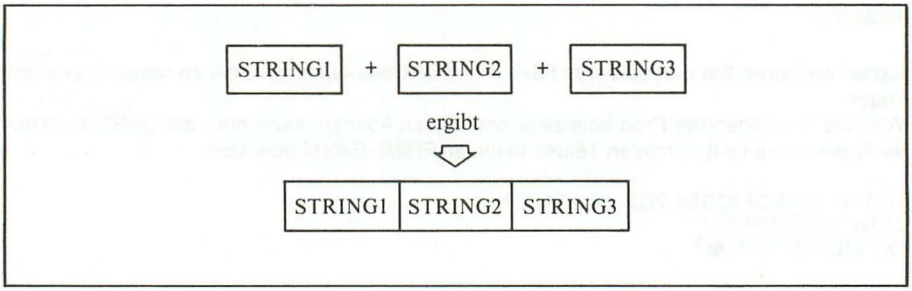


Abbildung 4-1: Zusammenfügen mehrerer Strings

Die einzige Begrenzung stellt die verfügbare Speicherkapazität dar. Im folgenden Beispiel sehen Sie, wie man mit Hilfe der LEN-Funktion Strings verbindet.

```
10 DIM A$(10),B$(10),C$(10)
20 A$="HAUS"
30 B$="SCHUH"
40 C$="REGAL"
50 A$(LEN(B$)+1)=B$
60 PRINT A$
70 B$(LEN(B$)+1)=C$
80 PRINT B$
90 END
RUN
```

```
HAUSSCHUH
SCUHREGAL

READY
```

Wenn Sie Strings nur für die Ausgabe zusammenfügen wollen, können Sie das auch ganz einfach mit PRINT und Semikolons erreichen. Vergleichen Sie die beiden Programme:

```
10 DIM A$(4),B$(5),C$(5)
20 A$="HAUS"
30 B$="SCHUH"
40 C$="REGAL"
50 PRINT A$;B$
60 PRINT B$;C$
70 END
RUN
HAUSSCHUH
SCHUHREGAL

READY
```

Diese Version erzeugt die gleiche Ausgabe wie die erste, enthält jedoch weniger Anweisungen. Deshalb ist das letzte Programm natürlich auch praktischer, es sei denn, Sie hätten vor, die zusammengefügten Strings im selben Programm wiederzubenutzen.

Graphikzeichen

Beim ATARI-Computer haben Sie 29 spezielle Graphikzeichen zur Verfügung, die Sie mit Hilfe der CTRL-Taste und in Verbindung mit anderen Tasten erzeugen. In Anhang D sind diese Funktionen aufgelistet. Sie können die Graphikzeichen, genau wie jedes andere Zeichen, auch Strings zuordnen. Wenn man die Funktion richtig einsetzt, kann man auf diese Weise Bilder entwerfen. In unserem Beispiel haben wir eine Spielkarte gezeichnet. Mit /(CTRL-F) und \(CTRL-G) zeichnet man die Ecken. Diesen Strich ¯(CTRL-M) benutzt man für die obere Kante der Karte, _(CTRL-N) für die untere. Das Zeichenl (CTRL-V) für die linke Seite, l(CTRL-B) für die rechte Seite und CTRL-; bildet das ♠ Zeichen.

```
100 ? "/¯¯¯¯¯¯¯¯¯¯¯¯¯\"
110 ? "l A             l"
120 ? "l               l"
130 ? "l               l"
140 ? "l               l"
150 ? "l               l"
160 ? "l               l"
170 ? "l       ♠       l"
180 ? "l               l"
190 ? "l               l"
200 ? "l               l"
210 ? "l               l"
220 ? "l               l"
230 ? "l             A l"
240 ? "\______________/"
```

Wie Sie sehen, enthalten einige der Programmzeilen die gleichen Anweisungen. Es wäre viel praktischer, eine Stringvariable zu bilden und diese zu veranlassen, den Inhalt von Zeile 120 wiederholt ausführen zu lassen.

Numerische Strings

Mit dem Inhalt eines numerischen Strings kann wie mit einer Zahl gerechnet werden. Die STR$ Funktion wandelt einen numerischen Wert in einen numerischen String um. Man kann davon ausgehen, daß ein numerischer Wert in der gleichen Form in einen String umgewandelt wird, wie er vom Computer bei der PRINT-Ausgabe dargestellt wird. Wenn die umzuwandelnde Zahl negativ ist, so ist das erste Zeichen des Strings, ein Minuszeichen. Ebenso verhält es sich mit sehr großen und sehr kleinen Zahlen. Stellt der ATARI-Computer den numerischen Wert in der Exponential Schreibweise dar, so wird dieser in der gleichen Form in den String übernommen. Diese Arbeitsweise demonstriert Ihnen das folgende Programm:

```
10 DIM N$(20)
20 ?"GEBEN SIE EINE ZAHL EIN";
30 INPUT N
40 N$=STR$(N)
50 ?"NUMERISCHER WERT ";N
60 ?"       STRING ";N$
69 REM MIT BREAK TASTE PROGR.BEENDEN
70 GOTO 20
RUN
GEBEN SIE EINE ZAHL EIN?-.0098
NUMERISCHER WERT -9.8E-03
            STRING -9.8E-03
GEBEN SIE EINE ZAHL EIN?1234567899
NUMERISCHER WERT 1234567890
            STRING 1234567890
GEBEN SIE EINE ZAHL EIN?
```

Starten Sie das Programm und geben Sie positive und negative Zahlen ein. Versuchen Sie es ebenso mit Zahlen, die mehr als zehn Ziffernstellen haben, sowie mit Dezimalzahlen.
Das nächste Beispiel zeigt wie man mit der LEN-Funktion numerische Strings aneinanderreiht.

```
10 DIM N$(100)
20 ?"GEBEN SIE EINIGE ZAHLEN EIN";
30 INPUT N
39 REM VERBINDE LETZTEN INPUT MIT N$
40 N$(LEN(N$)+1)=STR$(N)
50 ?"DIE ZAHL LAUTET JETZT:"
60 ?N$
70 ?
79 REM MIT BREAK TASTE PROGR. BEENDEN
80 GOTO 20
RUN
GEBEN SIE EINIGE ZAHLEN EIN?1234
DIE ZAHL LAUTET JETZT:
1234

GEBEN SIE EINIGE ZAHLEN EIN?5678
DIE ZAHL LAUTET JETZT:
12345678

GEBEN SIE EINIGE ZAHLEN EIN?9098
DIE ZAHL LAUTET JETZT:
123456789098

GEBEN SIE EINIGE ZAHLEN EIN?
```

Initialisierung von Stringvariablen

Es gibt einen Trick, mit dem man ganz bequem dem gesamten Inhalt einer Stringvariablen das gleiche Zeichen zuweisen kann. Dazu das folgende Beispiel:

```
10 DIM S$(100)
19 REM DIESES ZEICHEN WIRD S$ ZUGEORD.
20 S$="§"
29 REM DER LETZTE ZU DEFINIERENDE WERT
30 S$(100)=S$(1)
40 S$(2)=S$(1)
50 PRINT S$
60 END
RUN
§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§
§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§§
§§§§§§§§§§§§§§§§§§§§

READY
```

Zuerst weisen Sie dem ersten Zeichen des Strings einen Wert zu (Zeile 20). Dazu kann man jeden beliebigen Wert benutzen, auch in Verbindung mit der CHR$ Funktion. Als nächstes müssen Sie das letzte Zeichen angeben, dem "§" werden soll (Zeile 30). In diesem Fall ist das S$(100), da der String mit 100 dimensioniert wurde.
Der eigentliche Trick hierbei liegt in Zeile 40. Der Computer weist den Wert auf der rechten Seite des Gleichheitszeichens, dem Wert auf der linken Seite zu. So erhält S$(2), den Wert von S$(1). Nun geht das Programm zum nächsten Zeichen des Strings über und setzt S$(3) auf die linke und S$(2) auf die rechte Seite. Somit wird der neue Wert von S$(2) auch gleich S$(3) zugewiesen. Dieser Vorgang wiederholt sich nun Zeichen für Zeichen, wobei immer der String auf der linken Seite des Gleichheitszeichen, dem String auf der rechten Seite um einen Schritt voraus ist -S$(17)=S$(16)-. Die Zeile 40 ist beendet, wenn S$(100) den Wert von S$(99) erhält.

Simulierte String-Datenfelder

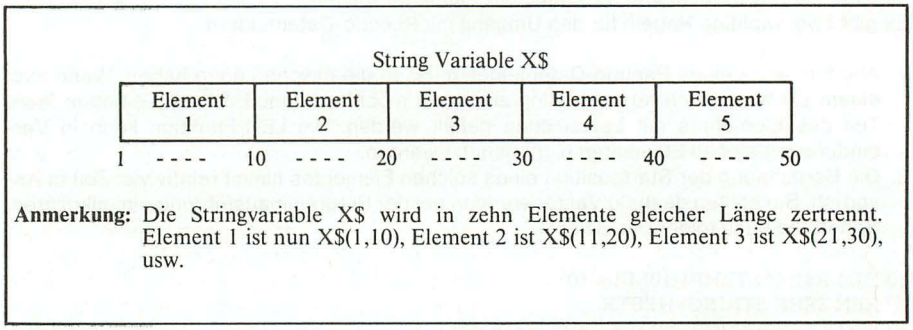
ATARI-BASIC erlaubt keine String-Datenfelder, jedoch kann man diese mit einer Stringvariablen simulieren. Das Ergebnis wird Pseudo-Datenfeld genannt. Um das zu erreichen, muß der gesamte String in Teilstrings von gleicher Länge zertrennt werden. Diese nun entstandenen Teilstrings stellen die Elemente des Pseudo-Datenfeldes dar (Abbildung 4-2). Um die Anfangsposition eines der Elemente zu berechnen, muß man seine Stellung innerhalb der Stringvariablen und die Länge der Elemente wissen.
Es gibt zwei wichtige Regeln für den Umgang mit Pseudo-Datenfeldern:

1. Alle Elemente eines Pseudo-Datenfeldes müssen die gleiche Länge haben. Wenn man einem Element einen kurzen String zuweisen möchte, so muß der verbleibende leere Teil des Elementes mit Leerzeichen gefüllt werden. Die LEN-Funktion kann in Verbindung mit diesen Elementen nicht benutzt werden.
2. Die Berechnung der Startposition eines solchen Elementes nimmt relativ viel Zeit in Anspruch. Sie sollten deshalb Verzögerungen bei der Programmausführung einkalkulieren. Sehen Sie dazu folgendes Beispiel:

```
10 DIM A$(100),TEMP$(10),BL$(10)
19 REM ZEHN STRING-WERTE
20 DATA LISA,BERND,SUSI,KLAUS,,1234568
901,+++,A+B=1E100,&§,ABRAKADABRA
```

```
29 REM BL$ WERDEN LEERZEICHEN ZUGEORD.
30 BL$=" ":BL$(10)=BL$:BL$(2)=BL$
39 REM LESEN DER WERTE
40 FOR N=1 TO 10
50 READ TEMP$
59 REM BERECHNE LAENGE DES GELESENEN W
ERTES
60 TL=LEN(TEMP$)
69 REM FUELLE KURZE WERTE MIT LEERZEIC
HEN AUF
70 IF TL<10 THEN TEMP$(TL+1)=BL$
79 REM BERECHNE ANFANG DER ELEMENTE
80 START=(N-1)*10+1
90 A$(START)=TEMP$
100 NEXT N
109 REM DARSTELLUNG DER DATEN
110 FOR N=1 TO 10
120 START=(N-1)*10+1
130 ?"ELEMENT ";N;" IST:";A$(START,STA
RT+9)
140 NEXT N
150 END
RUN
ELEMENT 1 IST:LISA
ELEMENT 2 IST:BERND
EIEMENT 3 IST:SUSI
ELEMENT 4 IST:KLAUS
ELEMENT 5 IST:
ELEMENT 6 IST:1234567890
ELEMENT 7 IST:+++
ELEMENT 8 IST:A+B=1E100
ELEMENT 9 IST:&5
ELEMENT 10 IST:ABRAKADABR

READY
```

In diesem Programm beherbergt der String A$ das Pseudo-Datenfeld. Es setzt sich aus zehn Elementen, die jeweils zehn Zeichen lang sind, zusammen (Zeile 10). Jedes dieser Elemente erhält einen Wert aus der DATA-Liste (Zeile 20). Wie Sie sehen wurden völlig unterschiedliche Daten eingegeben, unter anderem auch zwei, deren Länge 10 Zeichen überschreiten. In Zeile 30 werden der Stringvariablen BL$ Leerzeichen zugewiesen.



Anmerkung: Die Stringvariable X$ wird in zehn Elemente gleicher Länge zertrennt. Element 1 ist nun X$(1,10), Element 2 ist X$(11,20), Element 3 ist X$(21,30), usw.

Abbildung 4-2: Pseudo-Datenfeld

Nun liest das Programm die Werte aus der DATA-Liste und legt sie in TEMP$ ab (Zeile 50). Anschließend werden die Daten, die kürzer als zehn Zeichen sind, mit Leerzeichen zur vorgegebenen Länge aufgefüllt (Zeilen 60 und 70). Zum Schluß berechnet der Computer den Index von jedem Element des Pseudo-Datenfeldes (Zeilen 80 und 90).

Lassen Sie das Programm ablaufen und vergleichen Sie die anschliessende Darstellung mit den Werten der DATA-Liste.

Speicherung der Variablen

Die Höchstzahl der Variablen, die in einem ATARI-BASIC Programm benutzt werden können, ist 128. Hierbei wird jeder Name numerischer Variabler, Stringvariabler und Datenfelder gezählt. Zu beachten ist, daß ein Datenfeld nur einmal gerechnet wird, unabhängig von der Anzahl seiner Elemente.
ATARI-BASIC legt eine Variablennamen-Tabelle an, die bis zu 128 Namen aufnehmen kann. Wenn Sie in der direkten Betriebsart einen neuen Variablennamen verwenden, wird dieser sofort der Tabelle zugefügt. Bei Programmeingaben ist es etwas anders. Hier wird der Name erst registriert, wenn das Programm während der Ausführung mit den Variablen in Berührung kommt.

Die Variablennamen verbleiben in der Tabelle, bis der Befehl NEW eingetippt wird. Dann wird die vollständige Liste gelöscht. Es verändert den Inhalt der Tabelle also nicht, wenn Sie nur die der Variablen zugewiesenen Werte löschen.
Speichern Sie ein Programm auf der Kassette, so wird auch die Variablennamen-Tabelle mitgespeichert. Sobald Sie das Programm mit CLOAD wieder zurück in den Computer laden, wird die Tabelle, die vorher im Speicher war, gelöscht und durch die nun aktuelle, auf der Kassette gespeicherten, ersetzt.
In Kapitel 7 erfahren Sie, wie der Computer mit den Variablennamen-Tabellen verfährt, wenn die Programme auf einer Diskette gespeichert werden.

Direkter Speicherzugriff

Einige Befehle ermöglichen Ihnen den direkten Zugriff zum Speicher des ATARI-Computers, sowie zu seinen Datenkanälen der Ein- und Ausgabegeräte.
Sobald Ihre BASIC-Programme umfangreicher werden, benötigen Sie die Anweisungen für den direkten Speicherzugriff. In den Programmen dieses Kapitels werden Sie schon einige der Befehle kennenlernen; die folgenden Kapitel beschäftigen sich jedoch noch ausführlicher damit. Um zum Beispiel die weitreichenden Fähigkeiten des ATARI-Computers in seinem graphischen Bereich voll ausschöpfen zu können, müssen Sie vom direkten Zugriff Gebrauch machen.

Speicher und Adressierung

Der ATARI-Computer verfügt über 65536 Speicherplätze. Diese werden durch Adressenzahlen angesprochen, die sich von 0 bis 65535 erstrecken. Jeder verfügbare Speicherplatz ist so ausgelegt, daß er eine Zahl zwischen 0 und 255 beinhalten kann. Alle Daten, die im Speicher abgelegt werden, müssen zu Ziffern in diesem Zahlenbereich umgewandelt werden. Der ATARI-Computer bedient sich zur Umformung von Programmen und Daten in Zahlen, verschiedener Codeschemata. Für BASIC-Schlüsselworte benutzt er ein anderes Schema als für allgemeine Daten, numerische Werte, graphische Darstellungen, usw. Natürlich weiß der Computer auch, wie er die gespeicherten Daten wieder dechiffrieren muß. In seiner unbearbeiteten Fassung stellt sich also der Speicherinhalt in Form von

Zahlen zwischen 0 und 255 dar. Zum Entziffern der Zeichen im ATASCII-Code schlagen Sie in Anhang D nach.

PEEK und POKE

Mit der PEEK-Funktion läßt sich der Inhalt eines Speicherplatzes feststellen.
Dazu folgendes Beispiel:

```
10 A=PEEK(200)
```

Durch diesen Befehl wird der Inhalt der Speicherstelle 200 der Variablen A zugewiesen.
Mit dem Befehl POKE dagegen, legt man einen Wert in einer Speicherzelle ab. Die folgende Programmzeile bewirkt, daß der Wert der Variablen A im Speicherplatz mit der Adresse 8000 gespeichert wird.

```
20 POKE 8000,A
```

Die Adresse für PEEK oder POKE kann als Zahl, Variable oder Ausdruck angegeben werden. In jedem Fall darf die Angabe nicht kleiner als 0 und nicht größer als 65535 sein. Andernfalls gibt der Computer die Fehlermeldung 3 aus. Wird jedoch ein Speicherplatz angesprochen, der außerhalb der verfügbaren Speicherkapazität liegt, trotzdem erhalten Sie keine Fehlermeldung. Zum Beispiel: Ein ATARI-Computer, der mit 16K ausgerüstet ist, hat natürlich keinen Speicherplatz mit der Adresse 24000. In dem Fall wäre das Ansprechen dieser Adressenzahl für den Computer ohne Bedeutung und würde auch keinen Fehler verursachen.
PEEK liest den Inhalt einer Speicherzelle aus dem RAM-Bereich (Schreib-/Lesespeicher), sowie aus dem ROM-Bereich (Nurlesespeicher). Mit dem Befehl POKE dagegen, kann man nur den Inhalt von Speicherzellen im RAM-Bereich verändern.
In Anhang G finden Sie eine Liste mit häufig verwendeten Speicheradressen, die Sie mit PEEK und POKE ansprechen können.

Dateneingabe und Programmausgabe

Als Programmier-Anfänger entdeckt man schnell, daß die Ein/Ausgabe-Abschnitte eines Programms, die meisten Schwierigkeiten mit sich bringen. Nahezu jedes Programm benötigt Daten, die über die Tastatur eingegeben werden müssen. Dies wird durch zahlreiche INPUT-Befehle veranlaßt. Was geschieht aber, wenn der Benutzer aus Versehen die falsche Taste gedrückt hat oder wenn er, nachdem er schon einige weitere Daten eingegeben hat, feststellt, daß die ersten Angaben nicht korrekt sind? Ein gutes, ausgereiftes Programm muß berücksichtigen, daß der Bediener ein Mensch ist, dem deshalb auch immer wieder menschliche Fehler unterlaufen können.
Ebenso muß die Ausgabe der Ergebnisse vorher sorgfältig geplant werden. Da die Darstellung gut übersichtlich und verständlich sein muß, genügt es nicht, die bearbeiteten Daten mit ein paar wahllosen PRINT-Befehlen auszugeben. Daraus würde nämlich resultieren, daß einige Informationen falsch interpretiert oder völlig überlesen würden. In diesem Kapitel lernen Sie die vielfältigen Möglichkeiten des ATARI-Computers auszunutzen und somit die Bildschirmausgabe optimal zu gestalten.

Bildschirmausgabe

Der Prozess, mit dem man die Informationen auf dem Bildschirm so anordnet, daß sie leicht verständlich und außerdem auch optischen Ansprüchen gerecht werden, nennt man Bild-

schirmgestaltung. Hauptwerkzeug zum Darstellen der Daten ist natürlich der PRINT-Befehl. Er ist Ihnen durch die zahlreichen Beispiele schon vertraut.
Wie Sie wissen, beginnt die Ausgabe einer PRINT-Anweisung an der Stelle, an der sich der Cursor gerade befindet. Das bedeutet, daß der Cursor die Darstellung kontrolliert und steuert, wobei er im Normalfall nach jedem ausgegebenen Zeichen um eine Stelle nach rechts weiterrückt. Einige Zeichenkombinationen, wie Bildschirmkommandos oder Funktionstasten, führen den Cursor auch in andere Richtungen. Mit dem Befehl POSITION, kann man den Cursor auf jeden beliebigen Punkt des Bildschirms setzen. Nun wollen wir sehen, wie man mit diesen Kenntnissen die Bildschirmausgabe gestalten kann.

Zeilenvorschub

Wurde einer PRINT-Anweisung kein Komma oder Semikolon angefügt, dann wird deren Darstellung mit dem RETURN abgeschlossen. Das bewirkt, daß der Cursor an den Anfang der nächsten Bildschirmzeile springt. Die RETURN-Taste erzeugt das ATASCII EOL (end of line = Zeilenende) Zeichen, das einen Zeilenvorschub ausführt. Auch der PRINT-Befehl kann ein EOL-Zeichen erzeugen, das, wenn keine anderen Angaben gemacht wurden, an seiner letzten Stelle ausgegeben wird. Das erklärt auch, warum der Cursor nach Beendigung des PRINT-Befehls an den Anfang der nächsten Bildschirmzeile springt. Prüfen Sie nun das folgenden Programm, das jeweils an der ersten Position der 20 Bildschirmzeilen ein Z darstellt:

```
190 DIM C$(1)
200 C$="Z"
210 FOR I=1 TO 20
220 PRINT C$
230 NEXT I
240 PRINT "PUH!"
250 END
```

Natürlich könnte ein Komma oder ein Semikolon am Ende des PRINT-Befehls den Zeilenvorschub unterdrücken. Oder was meinen Sie dazu? Geben Sie nun diese kleine Variation ein:

```
190 DIM C$(1)
200 C$="Z"
210 FOR I=1 TO 760
220 PRINT C$;
230 NEXT I
240 PRINT "PUH!"
250 END
```

Wie Sie sehen, füllen sich zwanzig Zeilen des Bildschirms mit Z's. Aber woher kamen jetzt die zwanzig Zeilenvorschübe? Eigentlich sollte das Semikolon am Ende der Zeile 220 das EOL-Zeichen unterdrücken. Hat das in diesem Fall nichts genützt?
Wann immer etwas in der letzten Spalte einer Zeile dargestellt wird, löst das automatisch einen Zeilenvorschub aus. Dies ist eines der Merkmale des Bildschirms, das dafür sorgt, die Darstellung in der nächsten Zeile fortzusetzen, bevor der Ausdruck am rechten Rand nicht mehr zu erkennen ist.
Der Computer macht aber, außer den Cursor an den Anfang der nächsten Zeile zu setzen, noch etwas anderes. Er heftet eine vollständige Bildschirmzeile an das Ende der logischen Zeile. Diese Funktion kann man auch mit Kommas oder Semikolons nicht beeinflussen.
Buchstaben und Ziffern werden immer als solche dargestellt. Die Cursorsteuerungstasten ↓, ↑ ← und → (ATASCII-Code 28 bis 31), bewegen den Cursor immer auf die gleiche

Weise. Dagegen sind die Zeichen zum Löschen oder Einfügen von Zeilen (ATASCII-Code 156 und 157) nur in logischen, aber nicht in dargestellten Zeilen wirksam. Die TAB-Zeichen (ATASCII-Code 127, 158 und 159), deren Funktion wir anschließend besprechen werden, beeinflussen ebenfalls nur logische Zeilen. Wenn Sie solche Zeichen benutzen wollen, deren Funktion ausschließlich in logischen Zeilen wirksam ist, so vermeiden Sie am besten, diese ganz ans Ende der Zeile zu schreiben, um sicherzugehen, daß die logische Zeile nicht länger wird, als die dargestellte.
Im folgenden Programm wird demonstriert, wie durch einen Zeilenvorschub nach der letzten ausgeführten Anweisung, der ganze Bildschirminhalt nach oben geschoben und dadurch die oberste Zeile vom Bildschirm gerollt wird.

```
300 PRINT "DIESE ERSTE LOGISCHE ZEILE
BELEGT ZWEI BILDSCHIRMZEILEN
309 REM SPRINGE ZUR UNTERSTEN ZEILE
310 FOR N=1 TO 21
320 PRINT
330 NEXT N
340 FOR N=1 TO 37
350 PRINT "_";
360 NEXT N
369 REM KLINGEL
370 FOR N=1 TO 25
380 PRINT CHR$(253);
390 NEXT N
399 REM DARSTELLUNG IN DER LETZTEN SPA
LTE UND DER LETZTEN ZEILE
400 PRINT "5"
409 REM UNTERBRECHEN MIT BREAK TASTE
410 GOTO 410
```

Zuerst stellt das Programm die Meldung "DIESE ERSTE LOGISCHE ZEILE..." dar (Zeile 300). Dann werden 21 EOL-Zeichen ausgeführt, die die erste Zeile bis an den oberen Bildschirmrand schieben (Zeilen 310 bis 330). Anschließend werden 37 Bindestriche dargestellt, die den Cursor zur letzten Spalte in der letzten Reihe bewegen (Zeilen 340 bis 360). Nun ertönt der Konsolenlautsprecher für einige Sekunden (Zeilen 370 bis 390). In dieser Zeit können Sie beobachten, was mit der obersten Zeile passiert. Nachdem nämlich das allerletzte Zeichen dargestellt wurde (Zeile 400), veranlaßt der nachfolgende Zeilenvorschub, daß die vollständige erste logische Zeile (das sind hier zwei dargestellte Zeilen), vom Bildschirm geschoben wird. Der Cursor springt anschließend wieder an den Anfang der folgenden Zeile. Der letzte Befehl (Zeile 410) veranlaßt eine unendliche Schleife, die nur mit BREAK unterbrochen werden kann.
Wahrscheinlich haben Sie sich schon gewundert, weshalb der PRINT-Befehl in Zeile 380 keinen Zeilenvorschub ausgelöst hat. Das hat einen ganz einfachen Grund: Der ATASCII-Code 253 repräsentiert ein Klingelzeichen, also ein "nichtdruckendes" Zeichen. Deshalb nimmt es auch keinen Einfluß auf die Cursorposition.
Der automatische Zeilenvorschub signalisiert das Ende einer dargestellten Zeile, was nicht unbedingt mit dem Ende der logischen Zeile gleichzusetzen ist. Eine logische Zeile wird immer mit einem EOL-Zeichen beendet.

PRINT-Befehl und Komma

Daten, die in übersichtlichen Tabellen dargestellt werden, sind besser zu überblicken und zu verstehen. ATARI-BASIC bietet Ihnen zwei Möglichkeiten, den Ausdruck in dieser Form zu gestalten. Die eine ist, Kommas zwischen die einzelnen Werte der PRINT-Anweisung zu setzten, bei der anderen dagegen, wird die TAB-Taste benutzt.

```
9 REM BENZINKOSTENTABELLE
10 PRINT "PREIS PRO LITER";
20 INPUT PPL
30 PRINT "DURCHSCHN. KM PRO LITER";
40 INPUT KPL
50 PRINT "  KM","LITER","KOSTEN"
60 PRINT "----","-----","------"
70 FOR KM=100 TO 1700 STEP 100
79 REM BERECHNE LITER U. RUNDE AUF/AB
80 LIT=INT(KM/KPL*10)/10
89 REM BERECHNE KOSTEN UND RUNDE AUF/AB
90 KOST=INT(PPL*LIT*100)/100
100 PRINT KM,LIT,KOST
110 NEXT KM
120 PRINT
130 PRINT ,"KPL=";KPL,PPL;" DM PRO LITER"
140 END
```

Abbildung 4-3: Programmausdruck der Benzinkostentabelle

```
  KM        LITER      KOSTEN
----      -----      ------
 100         12.5       18.25
 200           25       36.50
 300         37.5       54.75
 400           50          73
 500         62.5       91.25
 600           75      109.50
 700         87.5      127.75
 800          100         146
 900        112.5      164.25
1000          125      182.50
1100        137.5      200.75
1200          150         219
1300        162.5      237.25
1400          175      255.50
1500        187.5      273.75
1600          200         292
1700        212.5      310.25

            KPL=8     1.46 DM PRO LITER

READY
```

Abbildung 4-4: Darstellung zum Benzinkostenprogramm (Abbildung 4-3)

Wenn der Computer auf ein Komma im PRINT-Befehl trifft, wird der Cursor nach rechts bis zur nächsten Standard-Druck-Position geschoben. Dabei werden zwischen dem letzten Zeichen des dargestellten Wertes und der jetzigen Cursorposition Leerzeichen eingefügt.

Jede Druck-Position umfaßt 10 Zeichen-Positionen. Die erste Standard-Druck-Position beginnt am linken Bildschirmrand in Spalte 1, die zweite beginnt in Spalte 11, die dritte in Spalte 21 und die vierte bei 31. Sehen Sie dazu das Beispielprogramm (Abbildung 4-3) das drei der vier Standard-Druck-Positionen belegt. Es gibt allerdings noch etwas im Gebrauch mit Kommas zu beachten. Die letzten zwei Zeichen vor der nächsten Druck-Position müssen frei sein. Andernfalls wird diese Druck-Position in der betreffenden Bildschirmzeile außer Kraft gesetzt und der Ausdruck bei der folgenden Druck-Position fortgeführt.

Im nächsten Beispiel hat das zweite Wort mehr als neun Buchstaben. Beachten Sie die Form der Darstellung:

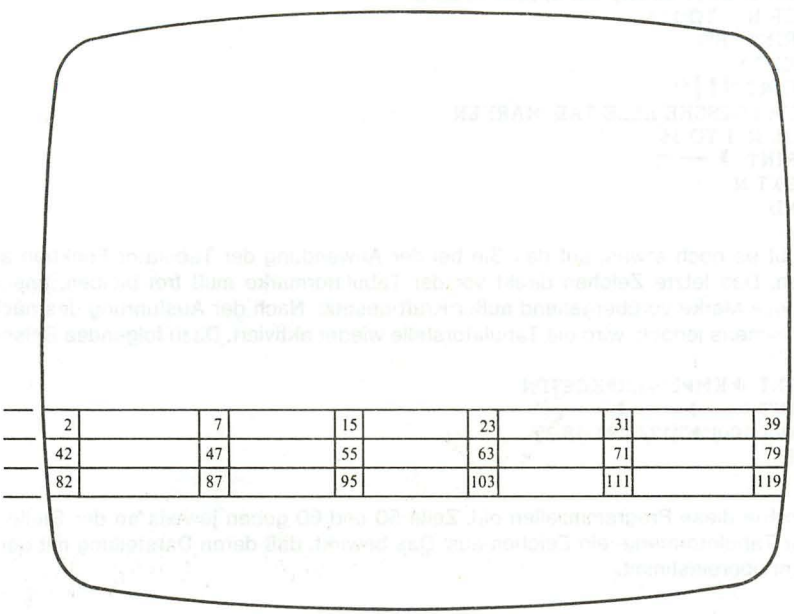
```
PRINT "STRASSE","HAUSNUMMER","WOHNORT"
STRASSE  HAUSNUMMER   WOHNORT

READY
```

Die Tabulator-Funktion

Die Tabulator-Funktion des ATARI-Computers arbeitet ähnlich wie die TAB-Taste einer Schreibmaschine. Sie ermöglicht Ihnen, den Cursor schnell zur nächsten, rechten Tabulatormarke zu bewegen. Nach dem Einschalten des Computers werden die Tabulatormarken automatisch auf die Spalten 7, 15, 23, 31 und 39 gesetzt (Abbildung 4-5). Die Funktion der TAB-Taste, läßt sich mit den Kommas eines PRINT-Befehls vergleichen. Jedoch arbeiten beide völlig unabhängig voneinander. Die Standard-Druck-Positionen haben keinen Einfluß auf die Tabulatormarken und umgekehrt.
Mit Hilfe der TAB-Taste kann man den Cursor von einer TAB-Marke zur nächsten bewegen, ohne daß der Bildschirminhalt, über den er springt, verändert wird. Drücken Sie die TAB-Taste, während der Cursor auf der letzten Marke steht, so springt er zum Anfang der nächsten logischen Zeile.
Die Tabulator-Funktion hat den ATASCII-Code 127. Um diese Funktion in Ihren Programmen einzubauen, können Sie entweder die CHR$-Funktion benutzen, oder aber, innerhalb eines Strings, ESC/TAB. Schreiben Sie nun das Programm von Abbildung 4-3 in der Form um, daß die Kommas durch TAB-Funktionen ersetzt werden. Das ► Zeichen stellt dabei die Funktionstasten ESC/TAB dar.

```
50 PRINT " ► KM►LITER►KOSTEN"
60 PRINT " ►---►-----►------"
100 PRINT "►";KM;"►";LIT;"►";KOSTEN
130 PRINT "►►KPL=";KPL;"►";PPL;" DM PRO LITER"
140 END
```

Das modifizierte Programm stellt die gleiche Tabelle dar, wie das Original (Abbildung 4-4), jedoch haben sich die Zwischenräume verändert.
Ebenso wie in der Programmierbetriebsart, kann man auch bei Direkteingaben zusätzliche Tabulatormarken setzen. Hierfür fahren Sie mit dem Cursor auf die gewünschte Position und drücken die Tasten SHIFT-TAB.
Auch mit einem PRINT-Befehl können Sie neue Tabulatormarken festsetzen. Zuerst muß dafür ein String dargestellt werden, der den Cursor auf die vorgesehene Position plaziert. Anschließend wird mit CHR$ (159) oder mit den Funktionstasten ESC/SHIFT-TAB, die neue Marke festgelegt.



Anmerkung: Die ersten zwei Spalten sind auf einigen Schirmen nicht sichtbar. Sie befinden sich außerhalb der linken Bildschirmbegrenzung.

Abbildung 4-5: Tabulatormarken bei der Standardeinstellung

Das folgende Programm setzt die neue Tabulatormarke in die fünfte Spalte und gibt anschließend eine Meldung aus, die an dieser Stelle beginnt:

```
110 PRINT "    ";CHR$(159)
120 PRINT CHR$(127);"DIESE MELDUNG WIRD IN DER FUENFTEN SPALTE
BEGONNEN"
RUN

    DIESE MELDUNG WIRD IN DER FUENFTEN SPALTE BEGONNEN

READY
```

Wollen Sie bei einer Direkteingabe eine Tabulatormarke löschen, so postieren Sie den Cursor auf der betreffenden Marke und drücken die Tasten CTRL-TAB.
Anders verhält es sich bei Programmeingaben. Hierbei setzen Sie den Cursor auf die Tabulatormarke und drücken die Funktionstasten ESC/CTRL-TAB oder geben die Funktion CHR$(158) ein.
Mit dem folgenden Programm werden alle vorhandenen Tabulatorstellen gelöscht. Wo ein ↑ steht, schreiben Sie ESC/CTRL- -, wo ▶ steht, drücken Sie ESC/TAB und für ←, drücken Sie ESC/CTRL-TAB.

```
498 REM ***LOESCHE ALLE TAB-MARKEN***
499 REM ZUERST SCHREIBE E. LANGE ZEILE
500 FOR N=1 TO 114
510 PRINT "H";
520 NEXT N
530 PRINT "↑↑↑";
539 REM LOESCHE ALLE TAB-MARKEN
540 FOR N=1 TO 16
550 PRINT "▶ ← ";
560 NEXT N
570 END
```

Nun gibt es noch etwas, auf das Sie bei der Anwendung der Tabulator-Funktion achten müssen. Das letzte Zeichen direkt vor der Tabulatormarke muß frei bleiben, ansonsten wird diese Marke vorübergehend außer Kraft gesetzt. Nach der Ausführung des nächsten EOL-Zeichens jedoch, wird die Tabulatorstelle wieder aktiviert. Dazu folgendes Beispiel:

```
50 PRINT "▶KM▶LITER▶KOSTEN"
60 PRINT "----▶-----▶------"
70 PRINT 100;"▶";12.5;"▶";18.25
80 END
```

Tippen Sie diese Programmzeilen ein. Zeile 50 und 60 geben jeweils an der Stelle direkt vor der Tabulatormarke ein Zeichen aus. Das bewirkt, daß deren Darstellung mit der Zeile 70 nicht übereinstimmt.

Formale Gestaltung der Bildschirmausgabe

Die beiden Möglichkeiten, die Sie bis jetzt kennengelernt haben, um die Ausgabe in Tabellenform zu gestalten, haben die einzelnen Datenspalten linksbündig dargestellt. Zahlen jedoch, sind einfacher zu lesen, wenn sie nach ihrem rechten Rand ausgerichtet werden. Das bedeutet, daß die jeweils letzten Ziffernstellen der Zahlen untereinander stehen. Die schon bekannte Benzinkostentabelle (Abbildung 4-3), haben wir dementsprechend umgeschrieben. Abbildung 4-6 zeigt die neue Version des Programms, in der folgende Variablen benutzt werden:

N	-	numerische Variable
NS	-	gibt den Platz an, der in einer Spalte verfügbar ist
BL$	-	dieser String enthält Leerzeichen
T$	-	Stringvariable, die nur zeitweise benutzt wird
N$	-	der Ausgabe-String

Wie Sie sehen, enthält das Hauptprogramm außer einigen anderen Veränderungen nun auch eine Unterroutine. In Zeile 5 werden die Variablen BL$, T$ und N$ dimensioniert. Anschließend wird BL$ mit Leerzeichen gefüllt (Zeile 7). Aus dem ein-zeiligen PRINT-Befehl des ursprünglichen Programms, sind nun drei Zeilen geworden (Zeilen 100, 102 und 104). Beachten Sie bitte, daß die Programmzeilen 100 und 102 mit Kommas beendet werden, um den Cursor an den linken Rand der nächsten Ausgabespalte zu postieren.

```
4 REM STRINGS FUER DAS UNTERPROGRAMM
5 DIM N$(10),T$(10),BL$(40)
6 REM BL$ WIRD MIT LEERZEICHEN GEFUEL.
7 BL$(1)=" ":BL$(40)=BL$:BL$(2)=BL$
9 REM BENZINKOSTENTABELLE
10 PRINT "PREIS PRO LITER";
20 INPUT PPL
30 PRINT "DURSCHN.KM PRO LITER";
40 INPUT KPL
50 PRINT "  KM","LITER","KOSTEN"
60 PRINT "----","-----","------"
70 FOR KM=100 TO 1700 STEP 100
79 REM BERECHNE LITER U. RUNDE AUF/AB
80 LIT=INT(KM/KPL*10)/10
89 REM BERECHNE KOSTEN U. RUNDE AUF/AB
90 KOST=INT(PPL*LIT*100)/100
100 NS=4:N=KM:GOSUB 11000:PRINT N$(1,N
S),
102 NS=5:N=LIT:GOSUB 11000:PRINT N$(1,
NS),
104 NS=6:N=KOSTEN:GOSUB 11000:PRINT N$
(1,NS)
110 NEXT KM
120 PRINT
130 PRINT ,"KPL=";KPL,PPL;" DM PRO LIT
ER"
140 END
10994 REM ****************************
10995 REM * UNTERPROGRAMM RICHTETE  *
10996 REM *   NUMERISCHE WERTE     *
10997 REM *     NACH RECHTS AUS    *
10998 REM ****************************
11000 T$=STR$(N)
11009 REM LOESCHE ALTE WERTE VON N$
11010 N$=BL$
11020 N$(NS-LEN(T$)+1,NS)=T$
11030 RETURN
```

Anmerkung: Die hervorgehobenen Zeilen unterscheiden sich vom Programm in Abbildung 4-3. Die Ausgabe wird in Abbildung 4-8 gezeigt.

Abbildung 4-6: Ausdruck des verbesserten Benzinkosten-Programs

Das Unterprogramm benötigt zur Berechnung der Ausgabepositionen, den Zugang zu jeder Ziffernstelle der einzelnen Zahlen. Dieser Prozeß ist in BASIC nur mit dem Inhalt von Stringvariablen möglich. Deshalb wird in Zeile 11000 die Zahl in einen numerischen String umgewandelt. Als nächstes wird der Ausgabestring mit Leerzeichen gefüllt. (Zeile 11010). Das garantiert, daß in den Teilen des Strings, die keine Ziffern enthalten, verläßliche Werte stehen. Zum Schluß wird die eigentliche Rechnung für die rechtsbündige Spaltendarstellung durchgeführt (Zeile 11020).
Hier wird herausgefunden, wieviel Ziffernstellen die Zahl hat und an welcher Position der Ausdruck beginnen muß, damit die Zahl mit der rechten Kante der Spalte abschließt (Abbildung 4-7).
Zur Übung ändern Sie das Programm so um, daß die Kommas durch die Tabulator-Funktionen ersetzt werden.

Sie werden sicherlich zustimmen, daß die Ausgabe in Abbildung 4-8 eine wirkliche Verbesserung zu der Abbildung 4-4 darstellt.
Noch übersichtlicher wäre es, wenn alle Zahlen, mit oder ohne Dezimalanteil anhand des Dezimalpunktes ausgerichtet werden würden. Dazu muß zuerst bestimmt werden, an welcher Stelle der einzelnen Spalten, der Punkt festgelegt wird. Anschließend muß jede Zahl nach der Position ihre Dezimalpunktes untersucht werden. Da BASIC mit Gleitkommazahlen arbeitet, könnte der Punkt also an jeder Stelle stehen. Wenn wir seine Position ermittelt haben, wird die ganze Zahl nach rechts oder links verschoben, um sie korrekt in die Tabellenspalte einzupassen. Bei einigen Werten kann das mit einer Kürzung der rechten Ziffernstellen verbunden sein, während bei sehr kurzen Zahlen zusätzliche Leerzeichen zugefügt werden. Sehen Sie dazu Abbildung 4-9. Das Haupt- und Unterprogramm wurde für diese Zwecke entsprechend verändert.

Der neuen Unterroutine wurden noch einige Funktionen hinzugefügt. Die Variabel DD zum Beispiel, spezifiziert die Anzahl der Ziffernstellen nach dem Dezimalpunkt. Die weiteren Variablen heißen: DP, NL und J.

Das Unterprogramm muß die Position des Dezimalpunktes in jeder einzelnen Zahl ausfindig machen. Es beginnt die Suche damit, daß es annimmt, der Punkt befände sich am Ende der Zahl (Zeile 11030). Daraufhin wird der String mit einer FOR-NEXT-Schleife Zeichen für Zeichen nach dem Dezimalpunkt abgetastet (Zeilen 11040 bis 11060). Wurde kein Punkt entdeckt, dann kehrt das Unterprogramm zu seiner ersten Vermutung zurück: Der Dezimalpunkt folgt der letzten Ziffernstelle. In diesem Moment enthält die Variable DP die Anzahl der Ziffernstellen bis und einschließlich dem Punkt.

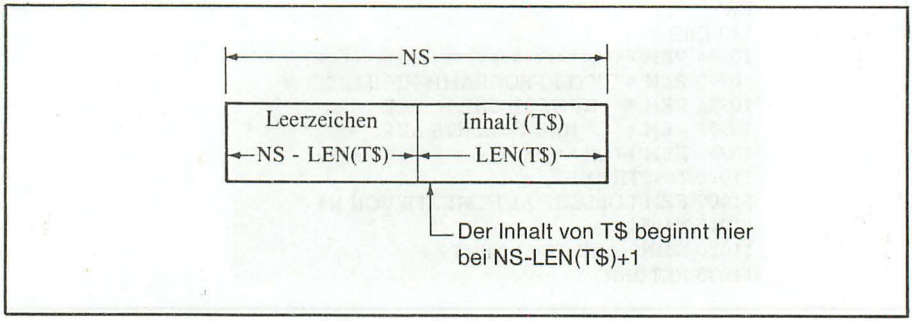


Abbildung 4-7: Rechtsbündige Darstellung eines Stringinhalts

Anschließend werden soviel Ziffernstellen dargestellt, wie in DP errechnet wurde, plus den Stellen, die in der Variablen DD abgelegt wurden.
Vergleichen Sie nun die Ausgabe dieser Programmversion (Abbildung 4-10) mit der des letzten Programms (Abbildung 4-8). Jetzt ist die ganze Tabelle noch leichter zu überblicken.
Wie Sie wissen, gibt BASIC am Ende von ganzen Zahlen keinen Dezimalpunkt aus. Ebenso werden keine "nachfolgenden Nullen" dargestellt. Das sind Nullen am Ende einer Zahl, die jedoch keinen Einfluß auf den Wert der Zahl haben. Falls diese nachfolgenden Nullen oder der abschließende Dezimalpunkt benötigt wird, so kann dies natürlich veranlaßt werden. Setzen Sie die untenstehenden Zeilen an das Ende des Programms von Abbildung 4-9:

```
11090 IF DD=0 THEN RETURN
11099 REM DEZIMALPUNKT VORHANDEN?
11100 N$(NS-DD,NS-DD)="."
11109 REM LEERZEICHEN WERDEN DURCH NUL
```

```
LEN ERSETZT
11110 FOR J=NS-DD+1 TO NS
11120 IF N$(J,J)=" " THEN N$(J,J)="0"
11130 NEXT J
11140 RETURN
```

```
  KM        LITER      KOSTEN
----        -----      ------
 100         12.5       18.25
 200         25         36.50
 300         37.5       54.75
 400         50         73
 500         62.5       91.25
 600         75        109.50
 700         87.5      127.75
 800        100        146
 900        112.5      164.25
1000        125        182.50
1100        137.5      200.75
1200        15         219
1300        162.5      237.25
1400        175        255.50
1500        187.5      273.75
1600        200        292
1700        212.5      310.25

            KPL=8      1.46 DM PRO LITER

READY
```

Abbildung 4-8: Ausgabe zum modifizierten Benzinkostenprogramm (Abbildung 4-6)

```
4 REM STRINGS FUER DAS UNTERPROGRAMM
5 DIM N$(10),T$(10),BL$(40)
6 REM BL$ WIRD MIT LEERZEICHEN GEFUEL.
7 BL$(1)=" ":BL$(40)=BL$:BL$(2)=BL$
9 REM BENZINKOSTENTABELLE
10 PRINT "PREIS PRO LITER";
20 INPUT PPL
30 PRINT "DURSCHN. KM PRO LITER";
40 INPUT KPL
50 PRINT "  KM","LITER","KOSTEN"
60 PRINT "-----","-----","------"
70 FOR KM=100 TO 1700 STEP 100
79 REM BERECHNE LITER U. RUNDE AUF/AB
```

Abbildung 4-9: Ausdruck des Benzinkosten-Programms. Werte werden anhand des Dezimalpunktes nach rechts ausgerichtet.

```
80 LIT=INT(KM/KPL*10)/10
89 REM BERECHNE KOSTEN U. RUNDE AUF/AB
90 KOST=INT(PPL*LIT*100)/100
100 NS=5:DD=0:N=KM:GOSUB 11000:PRINT N
$(1,NS),
102 NS=5:DD=1:N=LIT:GOSUB 11000:PRINT
N$(1,NS),
104 NS=6:DD=2:N=KOSTEN:GOSUB 11000:PRI
NT N$(1,NS)
110 NEXT KM
120 PRINT
130 PRINT ,"KPL=";KPL,PPL;" DM PRO LIT
ER"
140 END
10995 REM ***************************
10996 REM * UNTERPROGRAMM RICHTET   *
10997 REM *  NACH DEZIMALPUNKT AUS  *
10998 REM ***************************
11000 T$=STR$(N)
11009 REM LOESCHE ALTE WERTE VON N$
11010 N$=BL$
11030 DP=LEN(T$)+1
11039 REM SUCHE POSIT.D.DEZIMALPUNKTES
11040 FOR J=1 TO LEN(T$)
11050 IF T$(J,J)="." THEN DP=J:J=NS
11060 NEXT J
11069 REM BERECHNE DIE LAENGE DER ZAHL
11070 NL=DP+DD
11080 N$(NS-NL+1,NS)=T$
11090 RETURN
```

Anmerkung: Die hervorgehobenen Zeilen unterscheiden sich vom Programm in Abbildung 4-6. Die Ausgabe wird in Abbildung 4-10 gezeigt.

Abbildung 4-9: Ausdruck des Benzinkosten-Programms. Werte werden anhand des Dezimalpunktes nach rechts ausgerichtet.

Wie verhält sich das Programm, wenn man unrealistische Werte für den Benzinpreis oder die Kilometer eingibt? Tippen Sie 14.98 für den Literpreis und 2 für KM pro Liter ein. Der Computer stellt die Meldung 'ERROR 5 AT LINE 11080' dar, da die Kostenberechnung eine Zahl ergibt, die mehr Ziffernstellen hat, als die Tabellenspalte fassen kann. Nun haben Sie drei Möglichkeiten, diesen Fehler zu vereiteln: Das Unterprogramm oder das Hauptprogramm kann die Eingaben überprüfen, oder aber die Spalten werden von vornherein so breit gestaltet, daß auch die längste Zahl aufgenommen werden kann.
Den einfachsten Weg, so einem Fehler vorzubeugen, bietet die Längenüberprüfung durch das Unterprogramm. Fügen Sie diese Zeilen in das letzte Programm ein. Nun wird keine Fehlermeldung mehr erscheinen, ganz egal welche Werte anhand der Eingaben berechnet werden.

```
11020 IF ABS(N)>=10^(NS-DD-2) THEN N$=
T$:N$(NS)="*":RETURN
```

```
  KM        LITER       KOSTEN
 ----       -----       ------
 100        12.5        18.25
 200        25          36.50
 300        37.5        54.75
 400        50          73
 500        62.5        91.25
 600        75          109.50
 700        87.5        127.75
 800        100         146
 900        112.5       164.25
 1000       125         182.50
 1100       137.5       200.75
 1200       15          219
 1300       162.5       237.25
 1400       175         255.50
 1500       187.5       273.75
 1600       200         292
 1700       212.5       310.25

            KPL=8       1.46 DM PRO LITER

 READY
```

Abbildung 4-10: Ausgabe des letzten Benzinkostenprogramms (Abbildung 4-9).

Zeile 11020 veranlaßt, daß die betreffende Zahl in die Tabellenspalte paßt. Wenn der errechnete Wert sehr lang ist, erzeugt die Unterroutine soviel Ziffernstellen, wie die Spalte aufnehmen kann. In diesen Fällen gibt der Computer an der letzten Stelle ein Sternchen aus, um anzuzeigen, daß der normale Zahlenbereich überschritten wurde.
Lassen Sie das Programm nochmals ablaufen. Die Darstellung wird nun viel langsamer ausgeführt. Das resultiert aus dem Programmteil, der verhindert, daß die Ausgabe durch Fehlermeldungen gestoppt wird.

Cursorkontroll-Funktionen

Semikolons, Kommas und TAB-Funktionen eigneten sich gut für die Cursorkontrolle in den Beispielen, die wir bisher behandelt haben. Will man jedoch die Bildschirmausgabe noch etwas umfangreicher gestalten, so kann man sich weiterer Steuerfunktionen bedienen. ATARI-BASIC bietet Ihnen dazu vielseitige Möglichkeiten: zum Beispiel die CHR$-Funktion oder die Funktionstasten. Sehr häufig wird auch der POSITION-Befehl benutzt.

Löschen des Bildschirms

Oftmals muß während eines Programmablaufes der Bildschirm völlig gelöscht werden. Diese Funktion hat den ATASCII-Code 125 und setzt den Cursor nach dem Löschvorgang auf

seine Ausgangsposition (die linke, obere Bildschirmecke). Erzeugen kann man diesen Code mit CHR$(125), ESC/CTRL- < oder ESC/SHIFT- <.

Cursorsteuerungs-Funktionen

Man kann den Cursor mit Hilfe der Steuerungstasten ↑, ↓, ← und → auf jeden Punkt des Bildschirms postieren. Wie Sie bereits wissen, werden hierbei die Zeichen, über die der Cursor gleitet, nicht verändert. Die Funktionen sind bei der Programm- wie auch bei der Direkteingabe identisch.
Das Investitionsprogramm (Abbildung 4-11) berechnet den Gewinn, den Sie aus einer Investition nach einer bestimmten Periode erzielen. Nachdem der Computer das SPÄTERE GUTHABEN errechnet und dargestellt hat, steuert er den Cursor zurück zum ersten INPUT. Das ermöglicht Ihnen, völlig neue Daten einzugeben.

Position

Der POSITION-Befehl setzt den Cursor auf eine bestimmte Stelle des Bildschirms. Diese Position wird durch die Spalten- und Zeilenangabe spezifiziert. Sehen Sie dazu das folgende Beispiel:

```
9 REM BILDSCHIRM LOESCHEN
10 ?CHR$(125);
20 FOR J=1 TO 23
30 POSITION J,J
40 ?"*";
50 POSITION 24-J,J
60 ?"*";
70 NEXT J
79 REM MIT BREAK PROGRAMM UNTERBRECHEN
80 GOTO 80
```

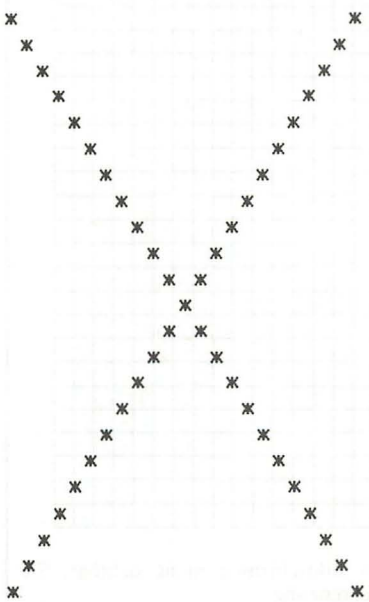
```
10 DIM R$(1),CU$(24),CR$(40)
19 REM STRINGS WERDEN MIT CURSORSTEUER
UNGSZEICHEN GEFUELLT
20 CU$=CHR$(28):CU$(24)=CU$:CU$(2)=CU$
30 CR$=CHR$(31):CR$(40)=CR$:CR$(2)=CR$
40 PRINT CHR$(125);"INVESTITIONS-ENTWI
CKLUNG"
50 PRINT
60 PRINT "KAPITALEINSATZ"
70 PRINT "NOMINALZINS"
80 PRINT "WIEOFT ZINSESZINSFESTLEGUNG
IM JAHR"
90 PRINT "LAUFZEIT IN JAHREN";CU$(1,4)
100 PRINT CR$(1,14);:INPUT KAP
110 PRINT CR$(1,11);:INPUT NZ
120 PRINT CR$(1,36);:INPUT ZZS
130 PRINT CR$(1,18);:INPUT JR
139 REM ZINSBERECHNUNG
140 IP=NZ/ZZS/100
```

Abbildung 4-11: Programm zur Investitions-Entwicklung

```
149 REM BERECHNE SPAETERES GUTHABEN
150 GHB=KAP*(1+IP)^(ZZS*JR)
160 PRINT "SPAETERES GUTHABEN: ";INT(G
HB*100+0.5)/100;" DM"
170 PRINT
180 PRINT "DATEN VERAENDERN";
190 INPUT R$
200 IF R$="J" THEN PRINT CU$(1,8);:GOT
O 100
210 END
RUN
INVESTITIONS-ENTWICKLUNG

KAPITALEINSATZ?6800
NOMINALZINS?9.5
WIEOFT ZINSESZINSFESTLEGUNG IM JAHR?4
LAUFZEIT IM JAHREN?10
SPAETERES GUTHABEN: 17388.64 DM

DATEN VERAENDERN?J
```

Abbildung 4-11: Programm zur Investitions-Entwicklung

Lassen Sie das Programm ablaufen. Zuerst wird der Bildschirm gelöscht und anschließend erscheint ein großes Kreuz.



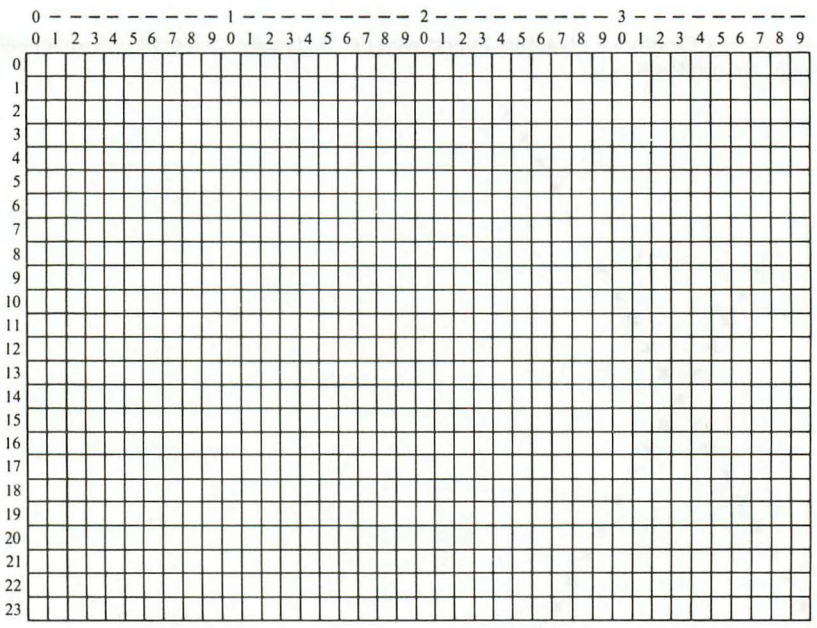
Haben Sie bemerkt, daß die äußerste linke und die oberste Zeile des Bildschirms frei geblieben sind? Bei der ersten Ausführung des POSITION-Befehls sollte der Cursor in die

Spalte 1 der ersten Reihe gesetzt werden. Genau das hat der Computer auch gemacht. In der Normaleinstellung hat man nur 38 Spalten zur Verfügung. Der Bildschirm ist jedoch in 40 Spalten aufgeteilt, wobei sich auf einem Standard-Bildschirm die ersten beiden Spalten außerhalb des linken Randes befinden. Der POSITION-Befehl ignoriert aber diese Randbegrenzung und berechnet den Bildschirm wie ein Rasternetz, das in der Normaleinstellung aus 40x24 Positionen besteht. Die Spalten werden von links nach rechts (von 0 bis 39) gezählt und die Zeilen von oben nach unten (von 0 bis 23). Sehen Sie dazu die Abbildung 4-12.

Zur Übung ersetzen Sie nun in dem Programm von Abbildung 4-11, die Cursorsteuerungs-Zeichen durch POSITION-Befehle. Anschließend löschen Sie die Stringvariablen CU$ und CR$, die nur für die Cursorsteuerung zuständig waren. Denken Sie auch daran, die Zeilen 100, 110, 120, 130 und 200 dementsprechend zu ändern.

Feststellen der Cursorposition

Mit dem POSITION-Befehl wird der Cursor nicht bewegt. Er aktualisiert vielmehr bestimmte Speicherplätze mit der neuen Cursorposition. Wenn jetzt etwas auf dem Bildschirm dargestellt wird, so erscheint es an der Position, die einer dieser Speicherplätze vorschreibt. Mit der PEEK-Funktion kann man jederzeit herausfinden, an welcher Stelle der Cursor als nächstes erscheinen wird. PEEK(84) enthält die neue Zeilenposition und PEEK(85) die neue Spaltenposition. In einigen Graphik-Modi, die Sie in Kapitel 8 und 9 kennenlernen werden, benötigt der ATARI-Computer zwei Speicherplätze für die Spaltenangabe.



Anmerkung: Die ersten zwei Spalten sind auf vielen Bildschirmen nicht sichtbar. Sie befinden sich außerhalb der linken Randbegrenzung.

Abbildung 4-12: Aufbau des Bildschirm-Rasternetzes

In diesem Fällen ergeben PEEK(86) * 256 + PEEK(85) die Spaltenposition. Jedes Mal, wenn durch eine PRINT-Anweisung etwas dargestellt wird, werden zwei weitere Speicherplätze mit der letzten Cursorposition aktualisiert. PEEK(91) teilt Ihnen die letzte Spaltenposition und PEEK(90) die letzte Zeilenposition mit.

Verändern der Bildschirmränder

Mit dem POKE-Befehl kann man die Bildschirmränder versetzen. Der ATARI-Computer hat den Speicherplatz 82 für den linken Rand und Platz 83 für den rechten Rand reserviert. In der Normaleinstellung ist der linke Rand auf Spalte 2 festgelegt. Mit dem folgenden Befehl läßt sich dieser auf Spalte 0 zurücksetzen:

```
POKE 82,0
```

In der Normaleinstellung befindet sich der rechte Bildschirmrand auf Spalte 39. Mit der nächsten Anweisung wird dieser auf Spalte 38 versetzt:

```
POKE 83,38
```

Wenn Sie die Bildschirmränder versetzt haben, dann denken Sie daran, daß sich der PRINT-Befehl genau nach den neuen Begrenzungen richtet. Erreicht die Ausgabe den rechten Rand, so springt der Cursor an den Anfang der nächsten Zeile. Selbst mit Steuerungstasten oder Kontrollfunktionen können Sie nicht erreichen, daß der Cursor außerhalb dieser Ränder erscheint. Testen Sie diesen Vorgang nun selbst:

```
10 POKE 83,10:REM RECHTER RAND
20 FOR J=1 TO 10
30 PRINT CHR$(31);:REM CURSOR N.RECHTS
40 NEXT J
50 PRINT "BLAU"
60 POKE 83,39
RUN
   BLAU

READY
```

Die Ausgabe startet an der Standardposition in Spalte 2. Der PRINT-Befehl innerhalb der FOR-NEXT-Schleife (Zeilen 20 bis 40) rückt den Cursor 9 mal bis zum rechten Rand (Spalte 10) vor. Danach springt der Cursor zurück zum linken Bildschirmrand, wo der Rest der Schleife ausgeführt wird. An der zweiten Stelle wird dann das Wort BLAU dargestellt. Der POSITION-Befehl bietet die einzige Möglichkeit, den Cursor außerhalb der festgelegten Ränder zu postieren. Stellt ihn das Programm rechts der rechten Bildschirmbegrenzung dar, so gibt der Computer nur das erste Zeichen des PRINT-Befehls aus und führt daraufhin sofort einen Zeilenvorschub durch. Der Rest des Wortes wird am Anfang der nächsten Zeile ausgegeben. Diesen Vorgang demonstriert Ihnen das folgende Beispiel:

```
10 POKE 83,10:REM RECHTER RAND
20 PRINT CHR$(125):REM SCHIRM LOESCHEN
30 POSITION 20,8
40 PRINT "VIOLETT"
50 POKE 83,39
```

Der Buchstabe "V" erscheint in Zeile 8, Spalte 20. Der übrige Inhalt der PRINT-Anweisung wird in Zeile 9 dargestellt, beginnend am linken Rand.
Werden die Bildschirmränder versetzt, so verändert sich auch die Länge der dargestellten, sowie der logischen Zeilen.

Regeln für den Aufbau von Eingabeprogrammen

Beinnahe alle Programme sind so konzipiert, daß der Anwender bestimmte Daten eingeben muß. Das Ziel eines jeden guten Programmes ist, mögliche Eingabefehler von vornherein auf ein Minimum zu reduzieren. Werden trotzdem Fehler gemacht, so sollen diese vom Anwender leicht zu erkennen und zu korrigieren sein. Die nächsten Seiten stellen Ihnen dazu folgenden Methoden vor:

- Darstellung von Anwenderhinweisen
- Rechnen Sie mit natürlichen, intuitiven Antworten
- Prüfen Sie die Eingaben auf Logik und Auswirkung auf den Programmablauf
- Verwenden Sie ein Unterprogamm, daß sich mit der Fehlerbehandlung auseinandersetzt
- Gruppieren Sie die INPUTs nach logischen Aspekten
- Ermöglichen Sie Überprüfungen und Korrekturen der Eingaben
- Verwenden Sie Steuerknüppel oder Drehregler zum Eingeben
- Beschränken Sie die Auswahlmöglichkeiten, indem Sie Menüs anbieten

Eingabeanweisungen

Mit den Eingabeanweisungen haben wir uns schon in Kapitel 3 beschäftigt. Wie unsere Beispielprogramme demonstrieren, sollten die Eingabeanweisungen möglichst kurz und bündig sein. Da der Platz auf dem Bildschirm meist knapp bemessen ist, sind wortreiche Erklärungen unpraktisch. Man sollte sich also auf das Wesentliche beschränken. Lassen Sie aber immer soviel Platz, daß die gesamte Antwort in eine Zeile paßt. Sollte das nicht möglich sein, dann stellen Sie die Eingabeanweisung in der einen Zeile dar und die Antwort in der nächsten. Da INPUT immer ein Fragezeichen ausgibt, ist es am besten, den vorhergehenden Satz auch als Frage zu formulieren.

Erweiterte Eingabe-Instruktionen

In manchen Fällen ist es unmöglich, eine Eingabe in kurzen Worten zufriedenstellend zu beschreiben. Entweder klingt es unverständlich oder aber die Instruktionen benötigen zu viel Platz. In diesem Fall kann man die ausführlichen Erklärungen irgendwoanders auf dem Schirm darstellen. Und so wird es gemacht. Das Programm gibt mit der Eingabeaufforderung (INPUT) eine Eingabeanweisung aus. Wenn der Anwender Probleme hat, kann der ein 'H' für Hilfe eingeben. Wenn er das getan hat, stellt das Programm, am besten in den untersten vier Bildschirmzeilen, erweiterte Eingabe-Instruktionen dar. Nachdem diese Erklärungen ausgegeben worden sind, kehrt das Programm zu der Stelle zurück, an der die Hilfestellung angefordert wurde.
Abbildung 4-13 demonstriert dazu ein Beispiel.
In diesem Programm wird nur der erste Buchstabe bestimmter Anweisungen eingegeben. Anschließend wird die ganze Anweisung in der obersten Zeile dargestellt. Ein komplettes Programm würde natürlich mehr tun, als nur das Wort auszugeben.
Abbildung 4-13 beinhaltet unter anderem eine merkwürdige Unterroutine in Zeile 32767, denn ihre einzige Aufgabe besteht darin, zurückzuverzweigen. Dieses

```
10 DIM BL$(40),BEF$(1),R$(1)
49 REM FUELLE BL$ MIT LEERZEICHEN
50 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
299 REM ERWEITERTE INSTR. IN ZEILE:
300 AMPSUB=2000
310 PRINT CHR$(125):REM SHIRM LOES.
320 POSITION 2,1
330 PRINT "BEFEHL (ABCDE, H FUR HILFE)
";
340 INPUT BEF$
360 POSITION 15,0
370 PRINT "BEFEHL LAUTET ";
380 ON ASC(BEF$)-64 GOSUB 400,600,800,
1000,1200,32767,32767,1400:GOTO 310
400 PRINT "ZUFUEGEN":FOR X=1 TO 300:NE
XT X:RETURN
600 PRINT "AENDERN":FOR X=1 TO 300:NEX
T X:RETURN
800 PRINT "LOESCHEN":FOR X=1 TO 300:NE
XT X:RETURN
1000 PRINT "DRUCKEN":FOR X=1 TO 300:NE
XT X:RETURN
1200 PRINT "ENDE":POSITION 2,20:END
1400 PRINT "HILFE"
1410 GOSUB 2000
1420 POP :GOTO 320
1994 REM ****************************
1995 REM *   UNTERPROGRAMM FUER   *
1996 REM *   EINGABEANWEISUNGEN   *
1997 REM ****************************
2000 GOSUB 2300
2009 REM DARSTELLUNG D. INSTRUKTIONEN
2010 PRINT "-------AUFSTELLUNG DER BEF
EHLE-------"
2020 PRINT
2030 PRINT " A=ZUFUEGEN B=AENDERN C
=LOESCHEN"
2040 PRINT " D=DRUCKEN  E=ENDE   H
=HILFE";
2050 RETURN
2296 REM ****************************
2297 REM * UNTERPROGRAMM Z. LOESCHEN *
2298 REM * D. INSTRUKTIONS-BEREICHES *
2299 REM ****************************
2300 POSITION 2,19
2310 FOR J=20 TO 23
2320 PRINT
2330 PRINT BL$(1,38);
2340 NEXT J
2350 POSITION 2,20
2360 RETURN
32767 RETURN:REM LEERE UNTERROUTINE
```

Abbildung 4-13: Ausdruck des Eingabe-Programms

Prinzip wird häufig in Verbindung mit dem ON-GOSUB Befehl angewendet. In diesem Programm zum Beispiel, ist kein 'F' vorgesehen. Wenn der Anwender nun ein F eintippt, so verzweigt das Programm zu der 'LEEREN UNTERROUTINE'. Das gleiche geschieht, wenn G eingegeben wird. Wird jedoch ein Buchstabe eingetippt, der im Alphabet nach H kommt, dann ist das Ergebnis des Ausdrucks im ON/-GOSUB-Befehl höher, als die Zeilennummer auf seiner Liste. In diesem Fall wird das Programm in Zeile 310 fortgeführt. Die Unterroutine von Zeile 1400 endet mit einem POP und GOTO-Befehl. Wenn dort anstatt dessen RETURN stünde, würde das Programm erst zur Zeile 380 und anschließend zu 310 zurückverzweigen. Dadurch würde der Schirm gelöscht werden, noch bevor man die erweiterten Instruktionen lesen könnte.

Man kann mit jedem Zeichen bewirken, daß Instruktionen dargestellt werden; in unserem Programm wählten wir das H. Diese Instruktionen können überall auf dem Bildschirm plaziert werden, es ist aber übersichtlicher, sie immer im gleichen Bereich ausgeben zu lassen.
Auf diese Weise haben Sie ausreichend Platz und können die Eingaben ganz ausführlich beschreiben. Bitte denken Sie daran, daß Sie dem Anwender genügend Zeit zum Lesen dieser Meldung lassen, bevor das Programm weiterläuft. Desgleichen sollten Sie die Möglichkeit schaffen, mit Hilfe eines INPUT-Befehls, die Instruktionsdarstellung zu unterbrechen und zu dem vorangegangenen Eingabeteil des Programms zurückzukehren. Ersetzen Sie nun das Unterprogramm von Zeile 2000 durch das, welches in Abbildung 4-14 gezeigt wird.

Eingabe-Masken

Viele Programme berücksichtigen die Input-Antworten nur bis zu einer bestimmten Länge. Das Programm könnte nun zum Beispiel einen String mit einer bestimmten Anzahl von Zeichen darstellen, die dem Anwender die maximale Antwortlänge demonstrieren. Ein derartiger String wird als Eingabe-Maske bezeichnet.
Für diese Masken kann man alle Zeichen verwenden, wie zum Beispiel:
Striche, Sternchen, Zahlen usw. Um dem Anwender gleich über die benötigte Datenart zu informieren, benutzt man ein bestimmtes Zeichen nur für Text Eingabe-Masken und ein anderes nur für numerische Eingabe-Masken. Im folgenden Programm haben wir Striche für Texteingaben und Sternchen für Zahlen verwendet:

```
10 DIM NMSK$(40),SMSK$(40),R$(40,CL$(4
0)
20 NMSK$="#":NMSK$(40)=NMSK$(1):NMSK$(
2):NMSK$(1)
30 SMSK$="_":SMSK$(40)=SMSK$(1):SMSK$(
2):SMSK$(1)
39 REM CL$ WIRD MIT CURSORSTEUERUNGS-Z
EICHEN GEFUELLT
40 CL$=CHR$(30):CL$(40)=CL$(1):CL$(2)=
CL$(1)
90 PRINT CHR$(125);"EINGABE DER BEGLIC
HENEN RECHNUNGEN"
100 PRINT "KENNZEICHEN DES RECHNUNGSAU
SZUGES IST ";SMSK$(1,6);CL$(1,7);
110 INPUT R$
120 PRINT "RECHNUNGSSUMME BETRAEGT ";N
MSK$(1,8);CL$(1,9);
130 INPUT R
140 END
```

Die beiden PRINT-Befehle, die die Eingabe-Masken darstellen, bewegen gleichzeitig auch den Cursor zurück zum Anfang der Maske (Zeilen 100 und 120).

Dort wird dann auch das INPUT-Fragezeichen ausgegeben (Zeilen 110 und 130).

```
1994 REM ****************************
1995 REM *    UNTERPROGRAMM FUER  *
1996 REM *    EINGABEANWEISUNGEN  *
1997 REM ****************************
2000 GOSUB 2300
2009 REM DARSTELLUNG D.INSTRUKTIONEN
2010 PRINT "-------AUFSTELLUNG DER BEF
EHLE--------"
2020 PRINT "NAEHERE EINZELHEITEN(ABCD,
N FUER NEIN)"
2030 PRINT "  A=ZUFUEGEN  B=AENDERN  C
=LOESCHEN"
2040 PRINT "  D=DRUCKEN   E=ENDE     H
=HILFE";
2049 REM NAEHERE EINZELHEITEN?
2050 POSITION 35,22
2060 INPUT R$
2070 IF R$="N" THEN RETURN
2080 ON ASC(R$)-64 GOTO 2150,2210,2210
,2100
2099 REM INSTRUKTIONEN FUER DRUCKEN
2100 GOSUB 2300
2110 PRINT "-----------BEFEHL:DRUCKEN-
----------"
2120 PRINT "   DRUCKT EINE LISTE DER
AUSGABE-"
2130 PRINT "KATEGORIEN UND IHRER NUMME
RN."
2140 RETURN
2149 REM INSTRUKTIONEN FUER ZUFUEGEN
2150 GOSUB 2300
2160 PRINT "-----------BEFEHL:ZUFUEGEN
-----------"
2170 PRINT "   UM EINE AUSGABE-KATEGO
RIE"
2180 PRINT "HINZUZUFUEGEN, MUESSEN SIE
EINE 5"
2190 PRINT "STELLIGE ZAHL UND EINE 1 B
IS 5"
2195 PRINT "STELLIGE BUCHSTABENBEZEICH
NUNG ANGEBEN"
2200 RETURN
2209 REM AENDERN UND LOESCHEN
2210 GOSUB 2300
2220 PRINT "----BEFEHL:AENDERN ODER LO
ESCHEN----"
```

Abbildung 4-14: Weiterentwickeltes Eingabe-Programm.

```
2230 PRINT "UM EINE AUSGABE-KATEGORIE
ZU AENDERN"
2240 PRINT "ODER ZU LOESCHEN, MUESSEN
SIE DIE"
2250 PRINT "NUMMER KENNEN. BENUTZEN SI
E DAZU DEN "
2255 PRINT "BEFEHL D."
2260 RETURN
```

Anmerkung: Die hervorgehobenen Zeilen unterscheiden sich vom Programm in Abbildung 4-13.

Abbildung 4-14: Weiterentwickeltes Eingabe-Programm.

Vorbereiten des INPUT-Befehls

Man kann die Fehlerhäufigkeit bei den Antworten des Anwenders verringern, indem man den INPUT sorgfältig vorbereitet. Ihr Programm sollte so konzipiert sein, daß es natürliche, intuitive Antworten des Anwenders verarbeiten kann. Es ist natürlich bequemer, Programme zu schreiben, die darauf basieren, daß der Benutzer alle Eingaben codiert. Das jedoch bereitet ihm unnötige Schwierigkeiten, da er bei jeder Programmbenutzung eine Art "mechanische Aufgabe" ausführen muß. Da der Computer aber solche Arbeiten hervorragend erledigt, sollten Sie ihm das Codieren überlassen. Wenn die logische Antwort ein Buchstabe oder ein Wort ist, das zur Programmfortführung in eine Zahl umgewandelt werden muß, dann veranlassen Sie einfach den Computer, diese Umwandlung zu erledigen.
Und genau das bewirken die Zeilen 310 bis 380 in Abbildung 4-13. Der Anwender gibt den ersten Buchstaben des Befehls ein (Zeile 340) und die Unterroutine findet anhand dieser Eingabe heraus, zu welcher Zeile das Programm daraufhin verzweigen soll. Es wäre natürlich viel problemloser das Programm so zu schreiben, daß gleich eine Zahl eingegeben werden muß, aber für den Anwender wäre es viel komplizierter.

Prüfen der Eingaben

Selbst bei sorgfältigster Konzipierung des INPUT-Befehl, können Sie nicht voraussehen, ob der Anwender eine korrekte Antwort eintippt. Deshalb sollten alle Eingaben vom Programm überprüft werden. Dazu gehört zum Beispiel, ob die Texteingabe zu lang ist, die Zahl noch innerhalb des angegebenen Bereichs liegt oder ob die Eingabe einen Fehler im weiterführenden Programm verursacht.
Wollen Sie ein gutes ausgereiftes Programm entwickeln, so sollten Sie alle Fehlermöglichkeiten einkalkulieren, die der Anwender machen könnte. Dabei muß das Programm Eingabefehler sofort erkennen und den Benutzer auffordern, korrekte Daten einzutippen.
ATARI-BASIC erleichtert Ihnen diese Arbeit, indem es von vornherein einige Eingabefehler abfängt. Zum Beispiel ist es nicht möglich, für numerische Variable, Texte einzugeben. Wie Sie wissen, gibt der Computer bei diesem Versuch eine Fehlermeldung aus und stoppt die Programmausführung.
Nun sind aber die diagnostischen Fähigkeiten des Computers begrenzt. Es ist

durchaus möglich, daß ein korrekter Wert eingegeben wird, der aber vom Programm nicht zu berechnen ist.

```
 10 DIM R$(1)
 40 GOSUB 300:REM INSTRUKTIONEN
290 END
300 PRINT CHR$(125);"INVESTITIONS-ENTW
ICKLUNG"
310 PRINT
320 PRINT "DIESES PROGRAMM BERECHNET D
EN"
330 PRINT "ZUKUENFTIGEN BETRAG EINER"
340 PRINT "INVESTITION, UNTER BERUECKS
ICHTIGUNG"
350 PRINT "DER ZINSEN. SIE MUESSEN DEN
"
360 PRINT "KAPITALEINSATZ, DEN NOMINAL
ZINS,"
370 PRINT "DIE JAEHRLICHEN ZINSESZINSF
ESTLEGUNGEN"
380 PRINT "UND DIE LAUFZEIT IN JAHREN
ANGEBEN."
385 PRINT
390 PRINT "VORAUSGESETZT, SIE TAETIGEN
 KEINE"
400 PRINT "ZUSAETZLICHEN ANZAHLUNGEN O
DER"
410 PRINT "ABHEBUNGEN, WIRD DIE"
420 PRINT "INVESTITIONS-ENTWICKLUNG NA
CH DIESER"
430 PRINT "FORMEL BERECHNET:"
435 PRINT
440 PRINT ,"GHB=KAP*(1+NZ/ZZS)^(ZZS*JR
)"
450 PRINT "WOBEI: GHB = SUMME NACH JR
JAHREN"
460 PRINT "     KAP = KAPITALEINSATZ
"
470 PRINT "      NZ = NOMINALZINS"
480 PRINT "     ZZS = ZINSESZINSFEST
LEGUNGEN"
490 PRINT "      JR = LAUFZEIT IN JA
HREN"
500 PRINT
510 PRINT "MIT RETURN STARTEN SIE DAS
PROGRAMM";
520 INPUT R$
530 PRINT CHR$(125);"INVESTITIONS-ENTW
ICKLUNG"
540 RETURN
100 INPUT X
200 PRINT 100/X
300 END
```

Abbildung 4-15: Programm zur Investitions-Entwicklung

In diesen Fällen verursacht die Eingabe erst im weiterführenden Programm einen Fehler. Probieren Sie das folgende Beispiel selbst aus:

Gibt man nach dem INPUT-Befehl (Zeile 100) die Zahl 0 ein, dann verursacht das in Zeile 200 einen Fehler, da die Division durch 0 nicht zu berechnen ist. Es ist jedoch ganz einfach, diesen Fehler zu unterbinden. Fügen Sie die folgenden Zeilen in das obenstehende Programm ein und starten Sie es erneut.

```
110 IF X<>0 THEN 200
120 PRINT "NICHT ZULAESSIG...BITTE KOR
REKTE EINGABE"
130 GOTO 100
```

Wie Sie feststellen konnten, ist es ohne großen Aufwand möglich, eine Eingabe zu überprüfen und gegebenenfalls weitere Schritte einzuleiten. Oftmals bieten sich hierfür der ON/GOTO- oder ON/GOSUB-Befehl an, die schneller arbeiten, als eine Reihe von IF/THEN-Befehlen.
In manchen Fällen nimmt die Fehlererkennung viel Programmierzeit und Speicherplatz in Anspruch. Stellen Sie die typische Ja-oderNein Frage, dann sind zehn Antworten möglich: ja, nein, JA, NEIN, Ja, Nein, J, n, J oder N. Diese alle einzeln zu überprüfen, wäre ein zu großer Aufwand, deshalb stellen wir Ihnen einen Weg vor, der diesen Prozeß erheblich beschleunigt.

```
10 DIM R$(40)
200 PRINT
210 PRINT "EINGABE WEITERER RECHNUNGEN
";
220 INPUT R$:R$=R$(1,1)
230 IF R$="J" OR R$="j" THEN 90
240 IF R$="N" OR R$="n" THEN END
250 GOTO 210
```

TRAP

ATARI-BASIC bietet Ihnen einen speziellen Befehl, der bei auftretenden Fehlern, die Darstellung der Fehlermeldung, sowie den Programmabbruch verhindert.

```
100 TRAP 20000
```

Sobald der Computer bei der Programmausführung auf einen Fehler stößt, bewirkt der Befehl eine Umleitung zu der Zeile, deren Zeilennummer hinter TRAP angegeben wurde. Die Fehlerkennzahl des aufgetretenen Fehlers findet man mit der Funktion: PEEK(195). Anhang A beschreibt, was die einzelnen Fehler-Code-Nummern bedeuten. ATARI-BASIC speichert ebenso die Zeilennummer, in der der Fehler auftrat.
Herausfinden kann man sie mit PEEK(187)x256+PEEK(186).
Der TRAP-Befehl muß nach jedem Fehlerfall erneut durchlaufen werden, damit er wieder wirksam wird. Um einen aktiven TRAP-Befehl unwirksam zu machen und den ATARI-Computer wieder zu seiner automatischen Fehlererkennung zu veranlassen, gibt man TRAP mit einer Zeilennummer zwischen 32767 und 65535 ein (z. B. TRAP 40000).

Unterprogramm zur Fehlerbehandlung

Für gewöhnlich wird der TRAP-Befehl immer im Zusammenhang mit einem Unterprogramm angewendet, das sich mit der Fehlerbehandlung beschäftigt. Im Fehlerfall veranlaßt ATARI-BASIC die Umleitung zu diesem Unterprogramm und nach dessen Ausführung, eine Rückverzweigung zu der angegebenen Zeilennummer. Im Normalfall ist das die Zeile, in der der Fehler auftrat. Ein Unterprogramm zur Fehlerbehandlung können Sie je nach Art des Fehlers, die verschiedensten Funktionen durchführen lassen.
Probieren Sie nun das folgenden Programm aus. Es enthält den TRAP-Befehl und eine Unterroutine, die die Fehlernummer und die Zeile, in der der Fehler auftrat, ausgibt. Bei einfachen Eingabefehlern wird um korrekte Daten gebeten.

```
10 DIM X$(10)
50 TRAP 8000
200 PRINT "GEBEN SIE BUCHSTABEN EIN";
210 INPUT X$
220 IF X$="E" THEN 500
230 PRINT "GEBEN SIE EINE ZAHL EIN";
240 INPUT X
249 REM FEHLER WENN EINGABE=0
250 A=X/X
499 REM PROGRAMMENDE
500 PRINT "DIE LETZTEN EINGABEN: ";X$;
" UND ";X
510 TRAP 40000:REM TRAP DEAKTIVIEREN
520 END
7998 REM +++++ FEHLERBEHANDLUNG +++++
7999 REM ERMITTLE FEHLERNUMMER
8000 E=PEEK(195)
8009 REM ZEILE IN D.D. FEHELR AUFTRAT
8010 EL=PEEK(187)*256+PEEK(186)
8020 IF E=3 OR E=8 THEN 8100
8030 PRINT "AHA! FEHLER NR. ";E;" GEFU
NDEN"
8040 PRINT "IN ZEILE NR. ";EL
8050 PRINT "NOTIEREN SIE SICH DIE FEHL
ERNUMMER"
8060 PRINT "UND IHRE EINGABEN."
8070 PRINT "SCHLAGEN SIE IM BENUTZERHA
NDBUCH NACH"
8080 END
8100 REM FEHLER BEI DER EINGABE
8110 PRINT CHR$(253);:REM KLINGEL
8120 PRINT "FEHLER...BITTE KORREKTE EI
NGABE"
8130 TRAP 8000
8140 GOTO EL
```

Eingabe-Hilfsprogramme

Jetzt kommen wir zu der Entwicklung einer allgemeinen Eingabe-Unterroutine, in der wir alle INPUT-Möglichkeiten, die wir bisher besprochen haben, verwenden: Eingabeanweisungen, erweiterte Eingabe-Instruktionen, Eingabe-Masken, Prüfen der Antworten, sowie ein Unterprogramm zur Fehlerbehandlung.
Das Eingabe-Unterprogramm bedient sich einiger anderer Unterroutinen. Eine davon hat die Aufgabe, bestimmte Bildschirmzeilen zu löschen (Abb 4-16). Mit Hilfe der Variablen

BL$ werden alle Zeichen einer Zeile, bis auf das jeweils letzte gelöscht. Würde man die letzte Spalte ebenfalls löschen, so würde dieses jedes Mal einen Zeilenvorschub auslösen und in der untersten Zeile veranlassen, daß der gesamte Bildschirminhalt um eine Zeile nach oben geschoben wird. Mit zusätzlichem Programmieraufwand ließe sich dieser Nebeneffekt abfangen, aber das würde sich natürlich auch auf die Ausführungsgeschwindigkeit auswirken und Speicherkapazität verbrauchen. In den meisten Fällen ist diese einfachere Version völlig ausreichend.
BL$ muß im Hauptprogramm dimensioniert und mit Leerzeichen gefüllt werden. Ein weiteres Hilfsprogramm bewirkt, daß die Fehlermeldung in der rechten, oberen Bildschirmecke ausgegeben wird (Abbildung 4-17). In der Stringvariablen ERM$ wird die Fehlermeldung abgelegt.

```
97 REM ******************************
98 REM * LOESCHE BILDSCHIRMZEILEN   *
99 REM ******************************
100 FOR J=L1 TO L2
110 POSITION 0,J
120 PRINT BL$(1,39);
130 NEXT J
140 RETURN
```

Abbildung 4-16. Programmausdruck zum Löschen von Bildschirmzeilen

```
167 REM ******************************
168 REM * DARSTELLUNG D.FEHLERMELDUNG*
169 REM ******************************
170 FOR J=1 TO 3
180 POSITION 20,0
190 PRINT ERM$;" FEHLER";CHR$(253);
200 FOR J1=1 TO 100:NEXT J1
210 POSITION 20,0
220 PRINT BL$(1,19)
230 NEXT J
240 RETURN
```

Abbildung 4-17. Programmausdruck zum Darstellen der Fehlermeldung

Das Unterprogramm veranlaßt, daß das Wort 'Fehler' neben der Fehlermeldung dargestellt wird. Außerdem ertönt bei jeder Fehlerdarstellung ein akustisches Signal. Die FOR-NEXT-Schleife (Zeile 200) bewirkt, daß die Meldung ein paar Sekunden auf dem Bildschirm stehenbleibt.
Wir benötigen jetzt noch eine Unterroutine, die den unteren Teil des Bildschirms, auf dem die erweiterten Instruktionen stehen, löscht (Abbildung 4-18). Die Funktion dieser Unterroutine besteht darin, ein anderes Unterprogramm aufzurufen, das einzelne Bildschirmbereiche löscht (Abbildung 4-16). Anschließend wird der Cursor auf die in Zeile 260 festgelegte Position gesetzt.
In Abbildung 4-19 sehen Sie nun den Ausdruck einer Eingabe-Unterroutine. In Zeile 600 wird die 'Fehlerbehandlung' aktiviert. Danach wird die Eingabe-Maske an vorher

definierten Positionen dargestellt (Zeilen 630 und 640). Da der INPUT mit einer Stringvariablen (Zeile 660) definiert wurde, ermöglicht das dem Anwender, statt einer Eingabe nur ein Fragezeichen einzutippen, welches dann die Darstellung erweiterter Instruktionen veranlaßt (Zeile 670). Drückt er jedoch nur die RETURN-Taste, ohne eine Eingabe zu tätigen, dann läuft das Programm so weiter, als hätte er eine 0 eingegeben (Zeile 700). In Zeile 720 wird die Eingabe bezüglich des Zahlenbereiches geprüft und in Zeile 750, bezüglich der Textlänge. In dieser Unterroutine wurden keine Längenbegrenzungen für numerische Eingaben berücksichtigt. Deshalb werden weder Ziffernstellen gekürzt, noch wird auf- oder abgerundet.
ERRHDL und AMPSUB (Zeilen 600 und 670) stellen in diesem Programm zwei variable Zeilennummern dar. Das ermöglicht dem aufrufenden Programm die Fehlerbehandlung und die Darstellung der erweiterten Instruktionen zu variieren. Die Zeilennummer die von AMPSUB vertreten wird, muß im Programm enthalten sein, auch wenn ihr Inhalt nur aus einem RETURN-Befehl besteht. Dagegen kann ERRHDL auch einen illegalen Wert, wie zum Beipiel 40000 haben. In diesem Fall wird TRAP unwirksam.
Nun benötigen wir noch eine Unterroutine, die den INPUT-Befehl mit Eingabeanweisungen darstellt. (Abbildung 4-20). Die Eingabeanweisung erscheint in der zweiten Bildschirmzeile. Anschließend wird die Eingabe-Unterroutine aufgerufen (Abbildung 4-19).
Das letzte Hilfsprogramm beschäftigt sich mit der Fehlerbehandlung (Abbildung 4-21). Macht der Anwender einen Eingabefehler, so wird die Unterroutine zur Darstellung der Fehlermeldung (Abbildung 4-17) aufgerufen und ausgeführt. Danach veranlaßt das Programm eine Rückkehr zum Anfang der Zeile, in der der Fehler auftrat. Ist der Fehler dadurch ausgelöst worden, weil keine Eingabe gemacht wurde, dann wird eine dementsprechende Information dargestellt und das Programm beendet.

```
247 REM ******************************
248 REM *LOESCHE BEREICH F.INSTUKTIO.*
249 REM ******************************
250 L1=20:L2=23:GOSUB 100
260 POSITION 2,20
270 RETURN
```

Anmerkung: Funktioniert nur in Verbindung mit dem Unterprogramm aus Abbildung 4-16.

Abbildung 4-18: Unterroutine zum Löschen des Instruktionsbereiches

```
596 REM ******************************
597 REM *  EINGABE - UNTERROUTINE   *
598 REM ******************************
600 TRAP ERRHDL
610 GOSUB 250
630 POSITION IC+1,IR
640 PRINT MSK$(1,IL)
650 POSITION IC,IR
659 REM PRUEFE EINGABE
660 INPUT R$
```

Abbildung 4-19: Allgemeine Eingabe-Unterroutine

```
663 REM FOR J1=1 TO IL
665 FOR J1=1 TO IL:IF R$(J1,J1)<>MSK$(
1,1) THEN NEXT J1
666 R$=R$(1,J1-1)
669 REM AUSFUEHRLICHE INSTRUKTIONEN?
670 IF R$="?" THEN GOSUB AMPSUB:GOTO 6
30
680 IF LO>HI THEN GOTO 750
690 REM WENN NICHTS EINGEGEBEN WIRD, I
ST R$=0
700 IF R$="" THEN R$="0"
710 R=VAL(R$)
720 IF R>=LO AND R<=HI THEN RETURN
729 REM FEHLER BEI ZAHLENREICHWEITE
730 ERM$="ZAHLENBEREICH"
740 GOSUB 170:GOTO 620
749 REM TEXT-EINGABE
750 IF LEN(R$)<=IL THEN RETURN
759 REM FEHLER BEI TEXTLAENGE
760 ERM$="TEXTLAENGE"
770 GOSUB 170:GOTO 620
```

Anmerkung: Läuft nur ordnungsgemäß mit folgenden Unterprogrammen: LOESCHE BILDSCHIRMZEILEN (Abb. 4-16), DARSTELLUNG D. FEHLERMELDUNG (Abb. 4-17) und FEHLERBEHANDLUNG (Abb. 4-21).

Abbildung 4-19: Allgemeine Eingabe-Unterroutine

In Tabelle 4-2 sind alle Hilfsprogramme gemäß ihren Zeilennummern aufgeführt. Außerdem zeigt sie die Variablen der einzelnen Unterroutinen und beschreibt, welche Unterprogramme nur in Verbindung mit anderen Unterprogrammen laufen. Wollen Sie ein Programm schreiben, daß eine oder mehrere Unterprogramme enthält, dann müssen die Stringvaria-

```
796 REM ******************************
797 REM *   EINGABEAUFFORDERUNG    *
798 REM ******************************
799 REM LOESCHE BILDSCHIRMBEREICH FUER
  EINGABEAUFFORDERUNG
800 L1=1:L2:GOSUB 100
809 REM STELLE EINGABEANWEISUNG DAR
810 POSITION 2,1
820 PRINT PRMT$
830 IC=2:IR=2:GOSUB 600
840 RETURN
```

Anmerkung: Dieses Programm benötigt die folgenden anderen Unterroutinen: LOESCHE BILDSCHIRMZEILEN (Abb. 4-16) und EINGABE-UNTERROUTINE (Abb. 4-19).

Abbildung 4-20: Programm zur Darstellung der Eingabeaufforderung

blen der betreffenden Hilfsprogramme dimensioniert werden, sowie den entsprechenden Variablen Werte zugewiesen werden (Tabelle 4-4).
Die Unterroutinen bauen darauf auf, daß die Stringvariablen im Hauptprogramm korrekt dimensioniert wurden.

```
7996 REM ++++++++++++++++++++++++++++++
7997 REM ++++++++ FEHLERBEHANDLUNG +++
7998 REM ++++++++++++++++++++++++++++++
7999 REM ERMITTLE FEHLERNUMMER
8000 E=PEEK(195)
8009 REM ZEILE IN D.D.FEHLER AUFTRAT
8010 EL=PEEK(187)*256+PEEK(186)
8020 IF E=3 OR E=8 THEN 8100
8030 PRINT "AHA!FEHLER NR. ";E;" GEFUN
DEN"
8040 PRINT "IN ZEILE NR. ";EL
8050 PRINT "NOTIEREN SIE SICH DIE FEHL
ERNUMMER"
8060 PRINT "UND IHRE DAZUGEHOERENDE EI
NGABE."
8070 PRINT "SCHLAGEN SIE IM BENUTZERHA
NDBUCH NACH"
8080 END
8100 REM FEHLER BEI DER EINGABE
8110 ERM$="EINGABE"
8120 GOSUB 170
8130 TRAP ERRHDL
8140 GOTO EL
```

Anmerkung: Benötigt die Unterroutine zur Darstellung der Fehlermeldung (Abbildung 4-17).

Abbildung 4-21: Programm zur Fehlerbehandlung

Tabelle 4-2: Aufstellung der Unterroutinen

Zeile	Bezeichnung	Variablen	Unterroutinen
100	4-16. Lösche Bildschirmzeilen	BL$, J, L1, L2	Keine
170	4-17. Darstellung der Fehlermeldung	BL$, ERM$, J, J1	Keine
250	4-18. Lösche Bereich für Instruktionen	L1, L2	100
600	4-19. Eingabe-Unterroutine	AMPSUB, ERM$, ERRHDL, HI, IC, IL, IR, J1, LO, MSK$, R, R$	ERRHDL, AMPSUB, 170, 250
640	4-37. String-Eingabe	IC, IL, IR, J, R, R$, X	Keine
800	4-20. Eingabeaufforderung	IC, IR, L1, L2, PRMT$	100, 160
850	4-38. Außerkraftsetzen der BREAK-Taste	J	Keine
6000	4-33. Bewegen des Cursors mit dem Steuerknüppel	BR, DLY1, J, LC, RC, SC, SR, TR	Keine
6500	4-31. Numerische Eingabe mit dem Steuerknüppel	BL$, HI, IC, IL, INC, IR, J, LO, R, SD	Keine
8000	4-21. Fehlerbehandlung	E, EL, ERM$, ERRHDL	170
8200	4-36. Eingabe des Datums	D, DAT$, IC, IR, M, MSK$, R$, J	8400
8400	4-36. Eingabe von Ziffern	J1, R, R( )	Keine

Tabelle 4-3: Dimensionierung der benötigten Stringvariablen*

Variable	Minimum	Maximum
BL$	39	keins
DAT$	8	8
ERM$	13	13
MSK$	0**	39
R$	1**	39
PRMT$	0	39

* Die Variablen finden Sie in den Abbildungen 4-16 bis 4-21, 4-31 und 4-36 bis 4-38.
** Muß sich nach dem längsten INPUT ausrichten.

Tabelle 4-4: Die Variablen mit ihren Verwendungszwecken

Variable	Änderung des Wertes?*	Verwendung
AMPSUB	Nein	Zeilennummer der Unterroutine, die erweiterte Instruktionen darstellt.
BL$	Nein	Leerzeichen um Löschen von Zeilen
BR	Nein	Die untere Zeilen-Begrenzung
DAT$	Ja	Eingegebenes Datum
DLYI	Nein	Geschwindigkeit des Cursors
E	Ja	Fehlernummer
EL	Ja	Nummer der Zeile, in der der Fehler auftrat
ERRHDL	Nein	Zeilennummer für die Fehlerbehandlung
HI**	Nein	Die höchste Zahl, die eingegeben werden kann
IC	Nein	Spaltenposition für die Eingaben, von 0 bis 38. 39 wäre auch möglich, verursacht aber einen Zeilenvorschub
IL	Nein	Länge der Eingabe
INC	Nein	Erhöhung um eine Einheit
IR	Nein	Zeilenposition für die Eingaben, von 0 bis 23
J	Ja	Eingegebenes Jahr
J1	Ja	Hilfsvariable
L1	Ja	Erste zu löschende Zeile, von 0 bis 23
L2	Ja	Letzte zu löschende Zeile, von L1 bis 23
LC	Nein	Die linke Spalten-Begrenzung
LO**	Nein	Die kleinste Zahl, die eingegeben werden kann
M	Ja	Eingegebener Monat
MSK$	Nein	Zeichen für die Eingabe-Maske
PRMT$	Nein	Eingabeanweisung
R	Ja	Übermittelt den numerischen Wert der Eingabe
R( )	Ja	Hilfs-Variable
R$	Ja	Übermittelt den String-Wert bzw. das String-Äquivalent der Eingabe
RC	Nein	Die rechte Spalten-Begrenzung
SC	Ja	Spaltenposition des Cursors, mit dem Steuerknüppel veränderbar
SD	Nein	Verzögerung während der Erhöhung
SR	Ja	Zeilenposition des Cursors, mit dem Steuerknüppel veränderbar
T	Ja	Eingegebener Tag
TR	Nein	Die obere Zeilen-Begrenzung

* Die Unterroutinen verändern die Werte nur bei den entsprechend gekennzeichneten Variablen.
** Wenn LO > HI, dann liest das Unterprogramm einen String-Inhalt ein. Andernfalls wird ein numerischer Wert eingelesen.

Man könnte das natürlich zum Beispiel bei den Variablen durch PRMT$ und MSK$ durch IL<=LEN(PRMT$) und IL<=LEN(MSK$) in den jeweiligen Hilfsprogrammen überprüfen, aber das würde unnötige Speicherkapazität und Zeit bei der Ausführung kosten. Da es sich bei einer falschen Dimensionierung um einen Programmierfehler und nicht um einen Anwenderfehler handelt, wird dieser, nach Entdeckung und Korrektur auch nicht mehr in Erscheinung treten.
Schreiben Sie nun ein Programm, das alle Unterroutinen (Abbildung 4-16 bis 4-21) enthält. Anschließend speichern Sie alles mit dem Befehl CSAVE auf einer Kassette ab. Auf diese Weise haben Sie diese Unterprogramme stets zur Verfügung und müssen sie bei Bedarf nicht erst neu eintippen. Danach laden Sie das Programm mit CLOAD zurück in den Speicher.
Geben Sie nun das in Abbildung 4-22 gezeigte Programm ein. Hier werden zuerst alle für die Unterroutinenen benötigten Variablen dimensioniert und initialisiert (Zeilen 10 bis 40).

In Zeile 90 wird die Verzweigung zum Hauptprogramm veranlaßt. In diesem wird zuerst die Länge der String-Eingabe auf maximal 20 Zeichen festgelegt (Zeilen 1010 bis 1040) und dann die numerische Eingabe vorbereitet (Zeilen 1050 bis 1110). Wie Sie feststellen können, enthält die Frage nach dem Punktstand schon die Eingabe, die gerade zuvor mit INPUT R$ (der Name), angefordert wurde (Zeilen 1050 bis 1070). Bei der ersten Eingabeaufforderung (Zeile 1010 bis 1040), wurde von der Möglichkeit, erweiterte Instruktionen zu erteilen, kein Gebrauch gemacht. Gelangt die Programmausführung zu Zeile 1030, so wird zu der "leeren" Unterroutine verzweigt. Dagegen werden für die numerische Eingabe, auf Anfrage, erweiterte Instruktionen ausgegeben (Zeilen 7000 bis 7030).

```
10 DIM PRMT$(60),MSK$(40),BL$(40),ERM$
(13),R$(20)
19 REM FUELLE BL$ MIT LEERZEICHEN
20 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
29 REM BEREITE EINGABE-MASKE VOR
30 MSK$="_":MSK$(40)=MSK$:MSK$(2)=MSK$
39 REM BEGINN DER FEHLERBEHANDLUNG
40 ERRHDL=8000
90 GOTO 1000
999 REM ****** HAUPT - PROGRAMM ******
1000 PRINT CHR$(125)
1009 REM TEXT-EINGABE
1010 PRMT$="NAME DES BOWLING-SPIELERS?
"
1020 IL=12:LO=1:HI=0
1030 AMPSUB=32767
1040 GOSUB 800
1049 REM ZAHLEN-EINGABE
1050 PRMT$="WELCHEN PUNKTSTAND HATTE "
1060 PRMT$(LEN(PRMT$)+1)=R$
1070 PRMT$(LEN(PRMT$)+1)="?"
1080 IL=7:LO=0:HI=300
1090 AMPSUB=7000
1100 GOSUB 800
1110 GOTO 1010
6997 REM INSTRUKT. ZUR ZAHLEN-EINGABE
```

Abbildung 4-22: Programm zur Eingabe des Punktestandes im Bowling-Turnier

```
6998 REM LOESCHE DEN DAFUER VORGESEHEN
BILDSCHIRMBEREICH
7000 GOSUB 250
7010 PRINT "GEBEN SIE EINEN POSITIVEN
WERT EIN,"
7020 PRINT "DER KLEINER ALS 300 IST."
7030 RETURN
32767 RETURN:REM LEERE UNTERROUTINE
```

Anmerkung: Mit diesem Programm können Sie die Unterroutinen 4-16 bis 4-21 ausprobieren.

Abbildung 4-22: Programm zur Eingabe des Punktestandes im Bowling-Turnier

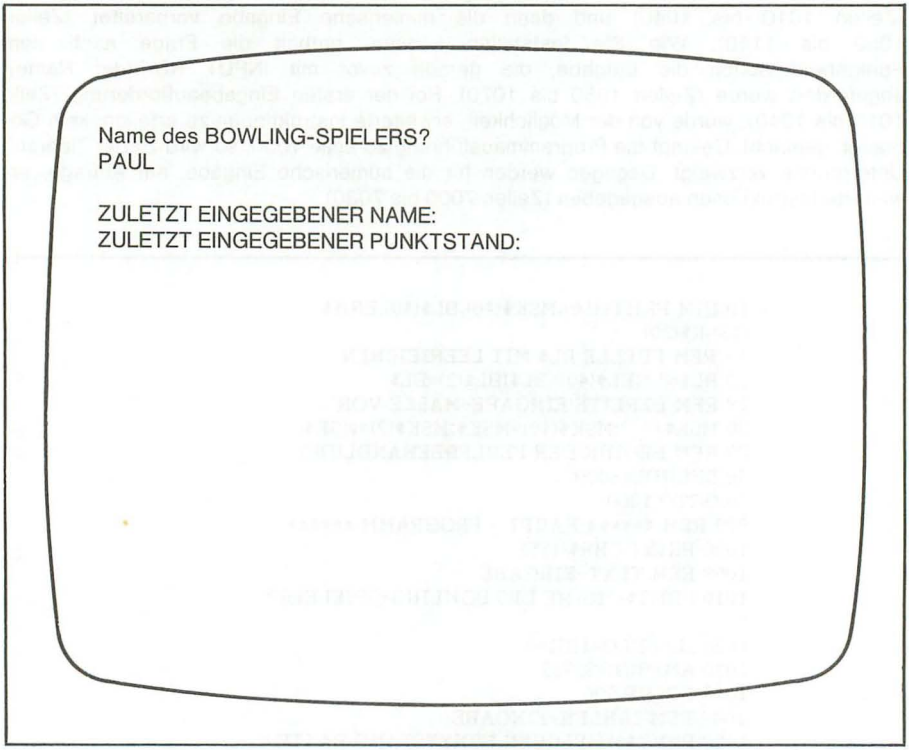


Abbildung 4-23: Darstellung der zuletzt eingegebenen Daten.

Eingaben von Datengruppen

In vielen Programmen müssen zahlreiche Daten eingegeben werden. Nun gibt es für den Aufbau des INPUT-Befehls verschiedene Möglichkeiten. Die eine ist, nacheinander die Eingabeaufforderungen auf dem Bildschirm darzustellen und die einge-

gebenen Daten einzeln zu verarbeiten. Nach diesem Prinzip ist auch das Beispielprogramm (Abbildung 4-22) aufgebaut, bei dem Namen und Punktzahlen separat angegeben werden.
In der Eingabeanweisung für den Punktestand, stellt das Programm auch den dazugehörenden Namen dar. Ohne diese Hilfestellung wäre die Eingabe ziemlich schwierig, da man sich immer den zuletzt eingetippten Namen merken müßte.
Versuchen Sie nun das Programm so umzuschreiben, daß der zuletzt eingegebene Name und Punktestand in der 5. und 6. Bildschirmzeile dargestellt werden (Abbildung 4-23).
Der beste Weg Daten einzugeben, die sich aus mehreren Einzelangaben zusammensetzen ist, den Bildschirm ähnlich wie ein Formular aufzubauen. Dabei wird berücksichtigt, daß die zusammengehörenden Einzelangaben solange dargestellt bleiben, bis die gesamte Dateneingabe abgeschlossen ist. Das Formular setzt sich aus einzelnen, numerierten Feldern zusammen, die ausreichend Platz für die Eingaben bieten. Die Daten werden gemäß den Feldnummern eingetippt (Abbildung 4-24).
Nehmen wir an, Sie möchten Namen und Adressen eingeben. Das Programm dafür ist einfacher, als Sie vielleicht denken. Die Dateneingabe besteht aus vier Einzelangaben: Name, Straße, Postleitzahl und Wohnort. Die Unterroutinen, die wir vorher besprochen haben, werden Ihnen den größten Teil der Arbeit abnehmen (siehe Tabellen 4-2, 4-3 und 4-4, sowie Abbildungen 4-16 bis 4-21). Laden Sie nun diese Hilfsprogramme wieder zurück in den Speicher Ihres ATARI-Computers.

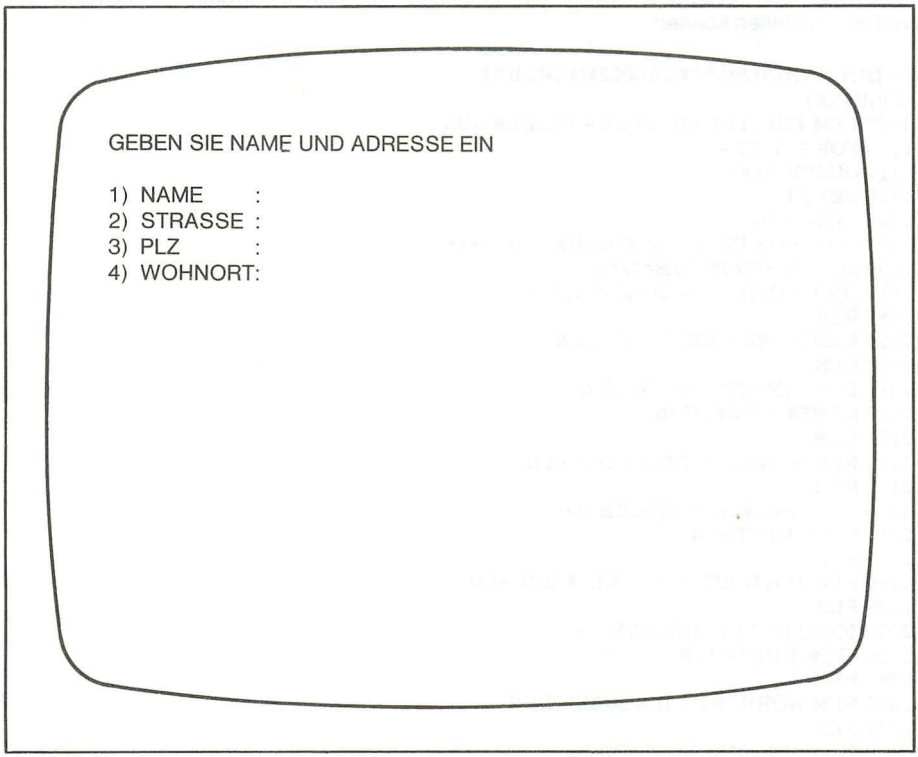


Abbildung 4-24: Darstellung eines Eingabeformulars

Vergewissern Sie sich, daß das Beipielprogramm von Abbildung 4-22 vollständig gelöscht ist. Nun fügen Sie die folgenden Programmzeilen hinzu, um das Eingabeformular auf dem Bildschirm darzustellen:

```
9 REM INITIALISIERUNG DER VARIABLEN
10 DIM PRMT$(40),MSK$(40),BL$(40),ERM$
(22)
19 REM FUELLE BL$ MIT LEERZEICHEN
20 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
29 REM EINGABE-MASKE WIRD VORBEREITET
30 MSK$="_":MSK$(40)=MSK$:MSK$(2)=MSK$
39 REM BEGINN DER FEHLERBEHANDLUNG
40 ERRHDL=8000
90 GOTO 1000
999 REM EINGABE-FORMULAR DARSTELLEN
1000 PRINT CHR$(125);"GEBEN SIE NAME U
ND ADRESSE EIN"
1010 PRINT:PRINT
1020 PRINT "1)  NAME:"
1030 PRINT "2)STRASSE:"
1040 PRINT "3)  PLZ:"
1050 PRINT "4)WOHNORT:"
1900 END
```

Nun muß das Programm noch die Eingaben für Name, Straße, Postleitzahl und Wohnort einlesen können:

```
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20)
1099 REM FUELLEN SIE ALLE 4 FELDER AUS
1100 FOR F=1 TO 4
1110 GOSUB 2000
1120 NEXT F
1180 GOTO 1100
1999 +++++++++ UNTERPROGRAMM 2000 ++++
2000 LO=1:HI=0:AMPSUB=32767
2010 ON F GOTO 2100,2200,2300,2400
2097 REM
2098 REM NAME - BIS 20 ZEICHEN
2099 REM
2100 IC=13:IR=3:IL=20:GOSUB 600
2110 NAME$=R$:RETURN
2197 REM
2198 REM STRASSE - BIS 20 ZEICHEN
2199 REM
2200 IC=13:IR=4:IL=20:GOSUB 600
2210 ST$=R$:RETURN
2297 REM
2298 REM POSTLEITZAHL - BIS 4 ZEICHEN
2299 REM
2300 IC=13:IR=5:IL=4:GOSUB 600
2310 PLZ$=R$:RETURN
2397 REM
2398 REM WOHNORT - BIS 20 ZEICHEN
2399 REM
2400 IC=13:IR=6:IL=20:GOSUB 600
2410 WOHN$=R$:RETURN
32767 RETURN:REM LEERE UNTERROUTINE
```

Geben Sie das Programm ein und lassen Sie es ablaufen. Sollten Fehler auftreten, so prüfen Sie den Programmausdruck noch einmal sorgfältig durch.
Dieses Namen- und Adressenprogramm stellt nacheinander für jedes einzelne Feld eine Eingabe-Maske dar. Dadurch wird dem Anwender auf einfachste Weise veranschaulicht, wo er die entsprechenden Daten eintippen soll.

Möglichkeiten zum Überprüfen und Ändern der Eingaben

Nach der abgeschlossenen Eingabe von Name und Adresse, sollte nun unser Programm den Anwender fragen, ob er irgendwelche Änderungen vornehmen möchte.
Wenn ja, muß die Nummer des zu ändernden Eingabefeldes angegeben werden. Anschließend verzweigt das Programm mit dem ON/GOSUB-Befehl, zu der betreffenden Unterroutine. In Abbildung 4-25 sehen Sie das vollständige Programm, bei dem nun auch die Möglichkeit zum Verändern der Eingaben berücksichtigt wurde (Zeilen 1130 bis 1230).
Prägen Sie sich den Programmaufbau sorgfältig ein. Achten Sie dabei besonders auf die Eingabehilfen, die in das Programm aufgenommen wurden. Dieses sind:

Dadurch, daß jedes Eingabefeld numeriert ist und daß nacheinander Eingabe-Masken dargestellt werden, weiß der Anwender sofort, welche Daten einzugeben sind und wieviel Zeichenstellen dafür zur Verfügung stehen.
Wenn die verfügbare Stellenanzahl überschritten wird, gibt das Programm eine Fehlermeldung aus.
Sobald der Anwender die Nummer des zu ändernden Feldes eingetippt hat, signalisiert ihm die erneute Darstellung der Eingabe-Maske, in welchem Feld das Programm die neuen Daten erwartet.
Die Antworten, die der Anwender auf Fragen, die das Programm stellt, eingibt, müssen nicht kodiert werden. Das Programm ist so aufgebaut, daß es ganz natürliche Antworten verarbeiten kann: J oder j für Ja, N oder n für Nein und die Nummern 1 bis 4 für die jeweiligen Felder.

Anschließend sehen Sie noch einige Anregungen für Eingabehilfen, die man ebenfalls noch in das Programm einbauen könnte:

Überprüfen der Postleitzahl auf unzulässige Symbole.
Beantwortet der Anwender die Frage nach Änderungen mit Nein, so könnte das Programm daraufhin den Satz: SIND SIE SICHER? darstellen. Wurde vorher versehentlich die falsche Taste gedrückt, so hat der Anwender nun eine zweite Möglichkeit zur korrekten Antwort.
Ferner könnte im Programm ein bestimmtes Zeichen berücksichtigt werden, mit dem die ursprüngliche Dateneingabe wieder hergestellt wird. Hat der Anwender nach der Änderungsfrage die falsche Feldnummer angegeben, so muß er in unserem Beispielprogramm die gleichen Daten noch einmal wiederholen. Dagegen ließe sich mit einem besonders definierten Zeichen Abhilfe schaffen.

Versuchen Sie nun selber, diese Eingabehilfen in das Namen- und Adressenprogramm einzufügen.

Eingaben mit Steuerknüppel

Jedes Programm, bei dem der Anwender mehrere Antwortmöglichkeiten hat, muß sehr sorgfältig vorbereitet werden. Unverständliche Eingabeanweisungen stellen potentielle Fehlerquellen dar. Das Programm muß so konzipiert sein, daß jede Eingabe genauestens

auf ihre Zulässigkeit überprüft wird, da fehlerhafte Dateneingaben den Programmablauf unterbrechen können.

```
9 REM INITIALISIERUNG DER VARIABLEN
10 DIM PRMT$(40),MSK$(40),BL$(40),ERM$
(22)
19 REM FUELLE BL$ MIT LEERZEICHEN
20 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
29 REM EINGABE- MASKE WIRD VORBEREITET
30 MSK$="_":MSK$(40)=MSK$:MSK$(2)=MSK$
39 REM BEGINN DER FEHLERBEHANDLUNG
40 ERRHDL=8000
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20)
90 GOTO 1000
97 REM ******************************
98 REM *  LOESCHE BILDSCHIRMZEILEN  *
99 REM ******************************
100 FOR J=L1 TO L2
110 POSITION 0,J
120 PRINT BL$(1,39);
130 NEXT J
140 RETURN
167 REM ******************************
168 REM * DARSTELLEN D. FEHLERMELDUNG*
169 REM ******************************
170 FOR J=1 TO 3
180 POSITION 11,0
190 PRINT ERM$;" FEHLER";CHR$(253);
200 FOR J1=1 TO 100:NEXT J1
210 POSITION 11,0
220 PRINT BL$(1,29)
230 NEXT J
240 RETURN
247 REM ******************************
248 REM *LOESCHE BEREICH F. INSTRUKT.*
249 REM ******************************
250 L1=20:L2=23:GOSUB 100
260 POSITION 2,20
270 RETURN
596 REM ******************************
597 REM *   EINGABE - UNTERROUTINE   *
598 REM ******************************
600 TRAP ERRHDL
620 GOSUB 250
630 POSITION IC+1,IR
640 PRINT MSK$(1,IL)
650 POSITION IC,IR
659 REM PRUEFE EINGABE
```

Anmerkung: Hier sind alle Hilfsprogramme enthalten, die wir bisher behandelt haben (Abb. 4-16 bis 4-21). Dieses Programm kann auch zu einer Adressenverwaltung umgeschrieben werden (siehe Abb. 5-2).

Abbildung 4-25: Name- und Adressenprogramm

```
660 INPUT R$
663 REM FOR J1=1 TO IL
665 FOR J1=1 TO IL:IF R$(J1,J1)<>MSK$(
1,1) THEN NEXT J1
666 R$=R$(1,J1-1)
669 REM AUSFUEHRLICHE INSTRUKTIONEN?
670 IF R$="?" THEN GOSUB AMPSUB:GOTO 6
30
680 IF LO>HI THEN GOTO 750
690 REM WENN NICHTS EINGEGEBEN WIRD, I
ST R$=0
700 IF R$="" THEN R$="0"
710 R=VAL(R$)
720 IF R>=LO AND R<=HI THEN RETURN
729 REM FEHLER BEI ZAHLENREICHWEITE
730 ERM$="ZAHLENBEREICH"
740 GOSUB 170:GOTO 620
749 REM TEXT-EINGABE
750 IF LEN(R$)<=IL THEN RETURN
759 REM FEHLER BEI TEXTLAENGE
760 ERM$="TEXTLAENGE"
770 GOSUB 170:GOTO 620
796 REM ******************************
797 REM *   EINGABEAUFFORDERUNG    *
798 REM ******************************
799 REM LOESCHE BILDSCHIRMBEREICH FUER
  EINGABEAUFFORDERUNG
800 L1=1:L2=2:GOSUB 100
809 REM STELLE EINGABEANWEISUNG DAR
810 POSITION 2,1
820 PRINT PRMT$
830 IC=2:IR=2:GOSUB 600
840 RETURN
999 REM EINGABE-FORMULAR DARSTELLEN
1000 PRINT CHR$(125);"GEBEN SIE NAME U
ND ADRESSE EIN"
1009 REM TEXT-EINGABE
1010 PRINT:PRINT
1020 PRINT "1)  NAME:"
1030 PRINT "2)STRASSE:"
1040 PRINT "3)   PLZ:"
1050 PRINT "4)WOHNORT:"
1099 REM FUELLEN SIE ALLE 4 FELDER AUS
1100 FOR F=1 TO 4
1110 GOSUB 2000
1120 NEXT F
1129 REM VERAENDERUNGEN
1130 PRMT$="WOLLEN SIE NOCH ETWAS VERA
ENDERN"
1140 LO=1:HI=0:AMPSUB=32767
```

Anmerkung: Hier sind alle Hilfsprogramme enthalten, die wir bisher behandelt haben (Abb. 4-16 bis 4-21). Dieses Programm kann auch zu einer Adressenverwaltung umgeschrieben werden (siehe Abb. 5-2).

Abbildung 4-25: Name- und Adressenprogramm (Fortsetzung)

```
1150 IL=1:GOSUB 800
1159 REM AUSWERTUNG DER ANTWORT
1160 IF R$="N" OR R$="n" THEN 1100
1170 IF R$="J" OR R$="j" THEN 1200
1180 ERM$="SCHREIBEN SIE J ODER N":GOS
UB 170
1190 GOTO 1130
1199 REM IN WELCHEM FELD?
1200 PRMT$="IN WELCHEM FELD"
1210 LO=1:HI=4:AMPSUB=32767
1220 IL=1:GOSUB 800
1230 F=R:GOSUB 2000:GOTO 1130
1900 END
1999 REM +++++ UNTERPROGRAMM 2000 ++++
2000 LO=1:HI=0:AMPSUB=32767
2010 ON F GOTO 2100,2200,2300,2400
2097 REM
2098 REM NAME - BIS 20 ZEICHEN
2099 REM
2100 IC=13:IR=3:IL=20:GOSUB 600
2110 NAME$=R$:RETURN
2197 REM
2198 REM STRASSE - BIS 20 ZEICHEN
2199 REM
2200 IC=13:IR=4:IL=20:GOSUB 600
2210 ST$=R$:RETURN
2297 REM
2298 REM POSTLEITZAHL - BIS 4 ZEICHEN
2299 REM
2300 IC=13:IR=5:IL4:GOSUB 600
2310 PLZ$=R$:RETURN
2397 REM
2398 REM WOHNORT - BIS 20 ZEICHEN
2399 REM
2400 IC=13:IR=6:IL=20:GOSUB 600
2410 WOHN$=R$:REUTRN
7996 REM ******************************
7997 REM *    FEHLERBEHANDLUNG     *
7998 REM ******************************
8000 E=PEEK(195)
8009 REM ZEILE IN D.D. FEHLER AUFTRAT
8010 EL=PEEK(187)*256+PEEK(186)
8020 IF E=3 OR E=8 THEN 8100
8030 PRINT "AHA!FEHLER NR. ";E;" GEFUN
DEN"
8040 PRINT "IN ZEILE NR. ";EL
8050 PRINT "NOTIEREN SIE SICH DIE FEHL
ERNUMMER"
8060 PRINT "UND IHRE DAZUGEHOERENDE EI
NGABE."
```

Anmerkung: Hier sind alle Hilfsprogramme enthalten, die wir bisher behandelt haben (Abb. 4-16 bis 4-21). Dieses Programm kann auch zu einer Adressenverwaltung umgeschrieben werden (siehe Abb. 5-2).

Abbildung 4-25: Name- und Adressenprogramm (Fortsetzung)

```
8070 PRINT "SCHLAGEN SIE IM BENUTZERHA
NDBUCH NACH"
8080 END
8100 REM FEHLER BEI DER EINGABE
8110 ERM$="EINGABE"
8120 GOSUB 170
8130 TRAP ERRHDL
8140 GOTO EL
32767 RETURN:LEERE UNTERROUTINE
```

Anmerkung: Hier sind alle Hilfsprogramme enthalten, die wir bisher behandelt haben (Abb. 4-16 bis 4-21). Dieses Programm kann auch zu einer Adressenverwaltung umgeschrieben werden (siehe Abb. 5-2).

Abbildung 4-25: Name- und Adressenprogramm

In vielen Fällen bietet sich hierfür als Lösung an, das Tastenfeld zu blockieren und den Steuerknüppel, bzw. den Drehregler, zur Eingabe zu benutzen. Natürlich ist das nicht bei allen Programmen möglich. Sie werden aber überrascht sein, wie vielseitig einsetzbar vor allem der Steuerknüppel ist.
Die STICK-Funktion findet die Stellung des Steuerknüppels heraus und gibt diese in Zahlen zwischen 5 und 15 weiter (Abbildung 4-26). STRICK dagegen, stellt fest, ob der Feuerknopf eines Steurknüppels gedrückt ist. Dieser Befehl

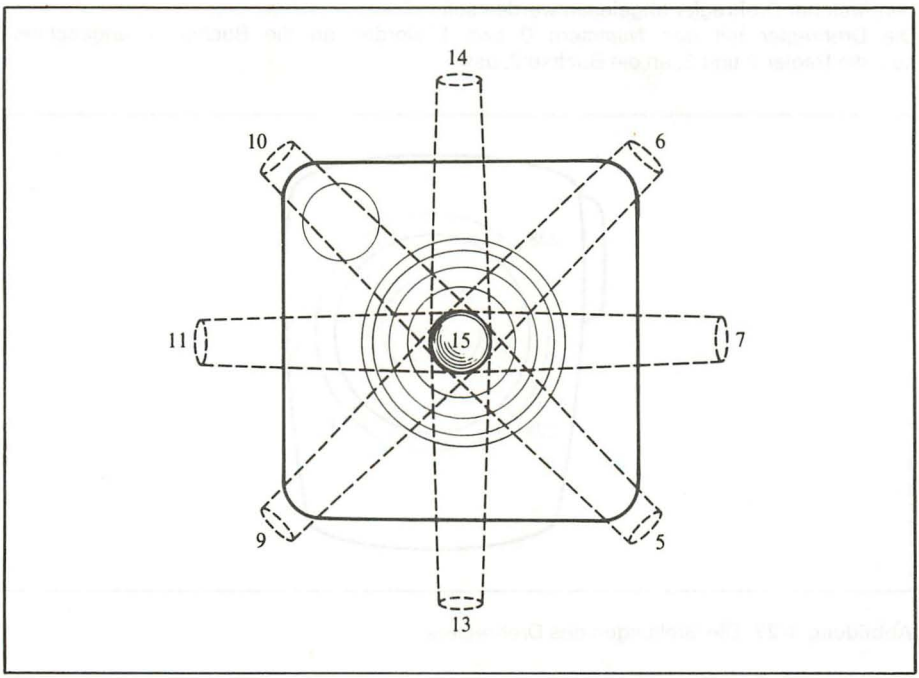


Abbildung 4-26: Die Stellungen des Steuerknüppels

gibt die Zahl 0 aus, wenn der Feuerknopf gedrückt ist und 1, wenn er nicht benutzt wird. Der ATARI Computer ermöglicht Ihnen, vier Steuerknüppel anzuschließen. Diese sind von 0 bis 3 numeriert. In den STICK und STRICK-Befehlen muß daher angegeben werden, an welchem der vier Knüppel die Werte abgelesen werden sollen.

Die Steuerknüppel werden an die vier Buchsen, die sich an der Vorderseite des Computers befinden, angeschlossen. Diese sind von links nach rechts numeriert (1 bis 4). Geben Sie dazu das folgende Programm ein:

```
10 PRINT CHR$(125)
20 POSITION 2,5
29 REM PROGRAMM MIT BREAK UNTERBRECHEN
30 PRINT "STEUERKNUEPPEL 0 STELLUNG: "
;STICK(0);" ";
40 PRINT "STEUERKNUEPPEL 0 FEUERKNOPF:
 ";
50 IF STRIG(0)=0 THEN PRINT "AN ";GOTO
 20
60 PRINT "AUS";GOTO 20
```

Die PADDLE-Funktion findet die Stellung des Drehknopfes bei einem Drehregler heraus und gibt diese in Zahlen zwischen 1 und 228 weiter (Abbildung 4-27). PTRIG dagegen, stellt fest, ob der Druckknopf eines Drehreglers gedrückt ist. Dieser Befehl gibt die Zahl 0 aus, wenn der Knopf gedrückt ist und 1 wenn er nicht benutzt wird. Drehregler werden jeweils paarweise angeschlossen. Der ATARI-Computer ermöglicht Ihnen, vier Paare, also acht Drehregler, zu installieren. Diese sind durch die Zahlen 0 bis 7 gekennzeichnet. Benutzen Sie die Befehle PADDLE oder PTRIG, so müssen Sie wiederum angeben, welcher Drehregler abgelesen werden soll.
Die Drehregler mit den Nummern 0 und 1 werden an die Buchse 1 angeschlossen, die Regler 2 und 3, an die Buchse 2, usw.

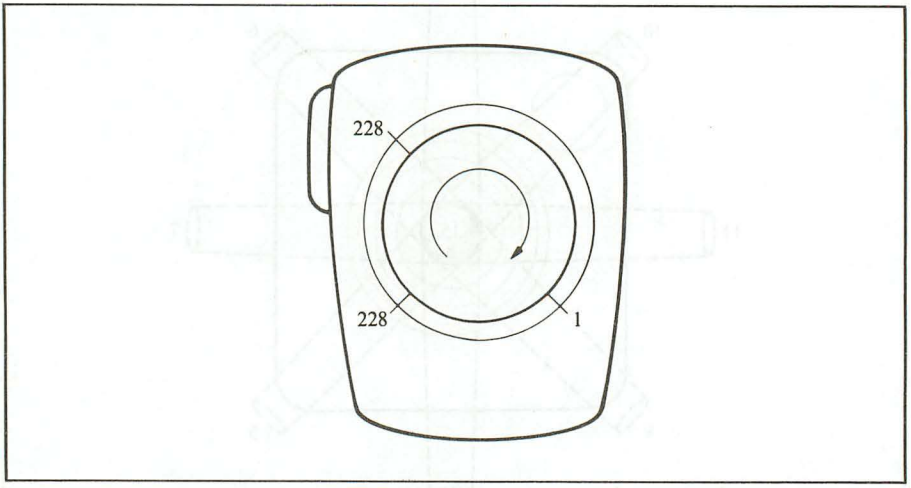


Abbildung 4-27: Die Stellungen des Drehreglers

Sehen Sie nun die Befehle PADDLE und PTRIG in einem praktischen Beispiel:

```
10 PRINT CHR$(125)
20 POSITION 2,5
29 REM PROGRAMM MIT BREAK UNTERBRECHEN
30 PRINT "DREHREGLER 0 STELLUNG:  ";
PADDLE(0);"  ";
40 PRINT "DREHREGLER 0 DRUCKKNOPF: ";
50 IF PTRIG(0)=0 THEN PRINT "AN ";GOTO
 20
60 PRINT "AUS";GOTO 20
```

Mit dem Steuerknüppel die Ausgabe regeln

Wenn Sie es mit umfangreichen Datenmengen zu tun haben, so ist es unmöglich diese alle zusammen auf dem Bildschirm darzustellen. Hierfür bietet sich an, die Daten in einzelne Teilkomplexe zu unterteilen und den Bildschirm wie eine Art Fenster zu benutzen, um zusammengehörende Datengruppen zu zeigen. Die Darstellung der Teilkomplexe ist einfach, wenn die Daten in Datenfeldern abgelegt worden sind. Stellen Sie sich nun vor, das Datenfeld wäre auf eine große Tafel geschrieben und Sie schauten darauf durch den Sucher einer Kamera. Die Tafel enthält sämtliche Daten, jedoch durch den Sucher können Sie jeweils nur einen kleinen Ausschnitt erkennen. Nun bewegen Sie den Sucher nach rechts, links, oben oder unten und können somit jeden Teil der Tafel nacheinander einsehen. Der Bildschirm stellt hierbei den Sucher dar, wobei man diesen mit dem Steuerknüppel bewegt.
Wir werden Ihnen diesen Vorgang nun anhand eines zwei-dimensionalen numerischen Datenfeldes demonstrieren. Jedes Element besteht aus einer vierstelligen Zahl und wird durch zwei Indizes gekennzeichnet.

```
X(i,j)=OiOj
```

Zum Beispiel:

```
X(3,2)=0302
X(19,8)=1908
X(11,12)=1112
```

Dieses numerische Datenfeld läßt sich am schnellsten mit einer verschachtelten FOR-NEXT-Schleife initialisieren.

```
10 DIM X(50,14)
49 REM DATENFELD INITIALISIEREN
50 PRINT CHR$(125);"EINEN MOMENT BITTE
";
60 FOR K=0 TO 14
70 FOR J=0 TO 50
80 X(J,K)=(J+1)*100+K+1
90 NEXT J
100 NEXT K
```

Um dieses Programm auszuführen, braucht der Computer ungefähr zehn Sekunden. Während dieser Wartezeit könnte der Anwender glauben, daß das Programm nicht läuft. Um zu verhindern, daß das Programm daraufhin neu gestartet wird, haben wir die Zeile 50 eingefügt. Sie sollten in allen Fällen, in denen für den Anwender nicht sofort erkenn-

bar ist, ob das Programm läuft, einen kleinen Hinweis darstellen.
Wie Sie aus der Abbildung 4-28 ersehen können, wurde nur ein Teil des Bildschirms verwendet. So läßt sich die Methode, bestimmte Datengruppen in einem sogenannten Fenster darzustellen, am besten demonstrieren. Man könnte dafür natürlich auch den gesamten Bildschirm benutzen, aber das hätte den Nachteil, daß man gleichzeitig keine anderen Daten darstellen könnte.
Nun wird eine Unterroutine eingefügt, die zuerst eine Tabelle mit den Überschriften: Spalte und Zei (Zeile) ausgibt. Geben Sie das Programm ein und lassen Sie es ablaufen.

```
998 REM +++++++++UNTERROUTINE+++++++++
999 REM DARSTELLUNG D. UEBERSCHRIFTEN
1000 FOR J=1 TO 3
1010 POSITION 3+J*10,3
1020 PRINT "SPALTE";
1030 NEXT J
1040 FOR J=0 TO 2
1050 POSITION 16+J*10,4
1060 IF C+J+1<10 THEN PRINT " ";
1070 PRINT C+J+1;
1080 NEXT J
1090 FOR J=0 TO 9
1100 POSITION 3,J+5
1110 PRINT "ZEI ";
1119 REM ZAHLEN SOLLEN RECHTSBUENDIG A
USGEGEBEN WERDEN
1120 IF R+J+1<10 THEN PRINT " ";
```

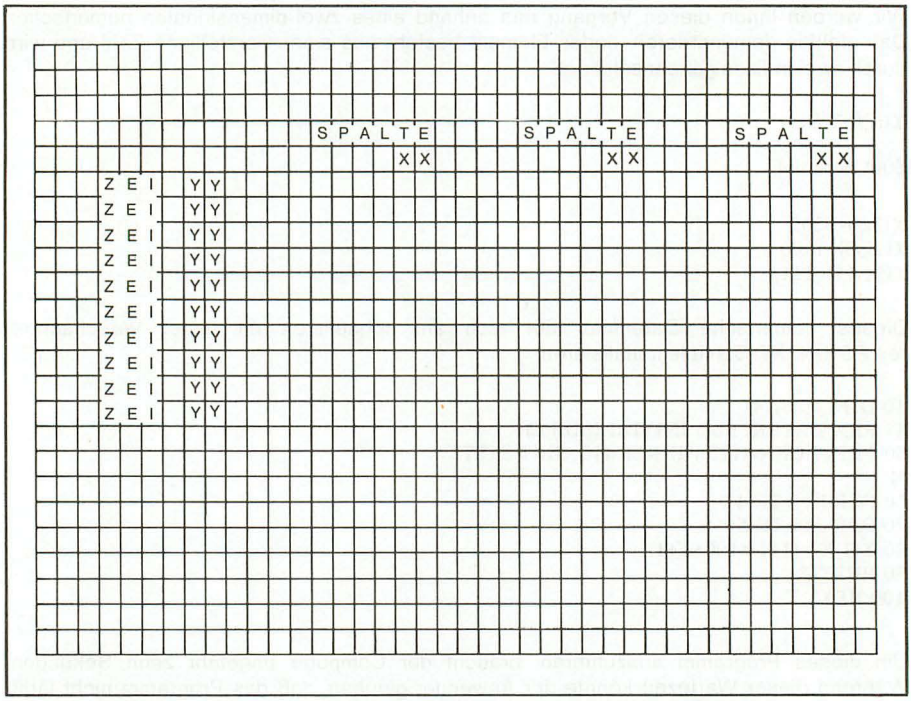


Abbildung 4-28: Darstellung eines Datenfensters auf dem Bildschirm

```
1130 PRINT R+J+1;
1140 NEXT J
1150 RETURN
```

Die folgenden Programmzeilen bewirken, daß der Wert der einzelnen Elemente des Datenfeldes ausgegeben wird. Das erste Element, das dargestellt wird, ist X(25,7) und erscheint in der oberen, linken Fensterecke.

```
20 DIM X$(10),BL$(40)
29 REM FUELLE BL$ MIT LEERZEICHEN
30 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
200 R=24:C=6
210 GOSUB 1000:REM UEBERSCHRIFTEN
220 FOR K=1 TO 10
230 FOR J=1 TO 3
240 POSITION J*10-1,K+4
250 X$=STR$(X(R+K-1,C+J-1))
260 PRINT BL$(1,10-LEN(X$));X$;
270 NEXT J
280 NEXT K
290 POSITION 2,1
350 END
```

In Programmzeile 250 wird jedes Datenfeld-Element in einen numerischen String umgewandelt. Auf diese Weise läßt sich die Ausgabe problemlos in Spalten darstellen (Zeile 260).
In diesem Programm wurde besonders darauf geachtet, daß die 40. Spalte in allen Zeilen frei bleibt. Die letzte Schreibstelle ist somit die 39. Benutzt man auch noch die 40. Spalte, so fügt der Computer automatisch eine neue Bildschirmzeile ein und setzt den Ausdruck in dieser Zeile fort. Unglücklicherweise würde das natürlich die ganze Tabelle verstören. Deshalb sollten Sie jede nur mögliche Vorsorge treffen, daß die automatischen Zeilenvorschübe und die von Ihnen programmierten, nicht aufeinandertreffen.
Dieser Programmteil liest sich die Stellung des Steuerknüppels ab:

```
299 REM SCHIEBE FENSTER NACH RECHTS
300 IF STICK(0)=7 AND C<12 THEN C=C+1:
GOTO 210
309 REM SCHIEBE FENSTER NACH LINKS
310 IF STICK(0)=11 AND C>0 THEN C=C-1:
GOTO 210
319 REM SCHIEBE FENSTER NACH UNTEN
320 IF STICK(0)=13 AND R<41 THEN R=R-1
:GOTO 210
329 REM SCHIEBE FENSTER NACH OBEN
330 IF STICK(0)=14 AND R>0 THEN R=R-1:
GOTO 210
339 REM PROGRAMM M. BREAK UNTERBRECHEN
340 GOTO 290
```

Wird der Steuerknüppel nun nach oben, unten, rechts oder links bewegt, so werden zuerst die Werte der Variablen R und C entsprechend korrigiert. Anschließend veranlaßt das Programm die Darstellung der angesteuerten, neuen Datenfeld-Elemente.
Den vollständigen Programmausdruck sehen Sie in Abbildung 4-29. Hierbei ließe sich noch vieles verbessern, zum Beispiel die sehr langsame Verarbeitungsgeschwindigkeit. Jedes Mal wenn die Spalten- oder Zeilennummern verändert werden, dauert es ca. zwei Sekunden, bis alle Daten im Fenster verändert sind. Das bedeutet, daß eine Verschiebung

um zehn Spalten 20 Sekunden dauert. Durch eine Effektivierung des Programms liesse sich die Verarbeitungssgeschwindigkeit wahrscheinlich um die Hälfte reduzieren, aber zehn Sekunden stellen immer noch einen zu langen Zeitraum dar. Anstatt einer ständigen Neudarstellung des Fensters, während der Steuerknüppel in eine Richtung gedrückt wird, wäre es vorteilhafter, dies nur durchzuführen, wenn der Steuerknüppel in Mittelstellung steht. Auf dieser Art wird das Fenster selbst bei einer größeren Verschiebung nur einmal neu dargestellt. Dadurch haben wir die Dauer einer Neudarstellung der Daten auf zwei Sekunden reduziert. Versuchen Sie nun diese Anregungen in nebenstehendem Programm umzusetzen.

```
10 DIM X(50,14)
20 DIM X$(4),BL$(40)
29 REM FUELLE BL$ MIT LEERZEICHEN
30 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
49 REM DATENFELD INITIALISIEREN
50 PRINT CHR$(125);"EINEN MOMENT BITTE
";
60 FOR K=0 TO 14
70 FOR J=0 TO 50
80 X(J,K)=(J+1)*100+K+1
90 NEXT J
100 NEXT K
199 REM HAUPTPROGRAMM
200 PRINT CHR$(125);"BEWEGEN SIE DAS F
ENSTER MIT DEM STEUERKNUEPPEL";
205 R=25:C=7
210 GOSUB 1000:REM UEBERSCHRIFTEN
220 FOR K=1 TO 10
230 FOR J=1 TO 3
240 POSITION J*10-1,K+4
250 X$=STR$(X(R+K-1,C+J-1))
260 PRINT BL$(1,10-LEN(X$));X$;
270 NEXT J
280 NEXT K
290 POSITION 2,1
299 REM UEBERPRUEFE STELLUNG DES STEUE
RKNUEPPELS
300 IF STICK(0)=7 AND C<12 THEN C=C+1:
GOTO 210
310 IF STICK(0)=11 AND C>0 THEN C=C-1:
GOTO 210
320 IF STICK(0)=13 AND R<41 THEN R=R+1
:GOTO 210
330 IF STICK(0)=14 AND R>0 THEN R=R-1:
GOTO 210
340 GOTO 290
998 REM +++++UNTERROUTINE 1000++++++++
999 REM DARSTELLUNG D. UEBERSCHRIFTEN
1000 FOR J=1 TO 3
1010 POSITION 3+J*10,3
1020 PRINT "SPALTE";
1030 NEXT J
1040 FOR J=0 TO 2
```

Abbildung 4-29: Programm zum Verschieben des Datenfensters

```
1050 POSTION 16+J*10,4
1059 REM ZAHLEN SOLLEN RECHTSBUENDIG A
USGEGEBEN WERDEN
1060 IF C+J+1<10 THEN PRINT " ";
1070 PRINT C+J+1;
1080 NEXT J
1090 FOR J=0 TO 9
1100 POSITION 3,J+5
1110 PRINT "ZEI ";
1119 REM ZAHLEN SOLLEN RECHTSBUENDIG A
USGEGEBEN WERDEN
1120 IF R+J+1<10 THEN PRINT " ";
1130 PRINT R+J+1;
1140 NEXT J
1150 RETURN
```

Abbildung 4-29: Programm zur Darstellung eines Datenfensters

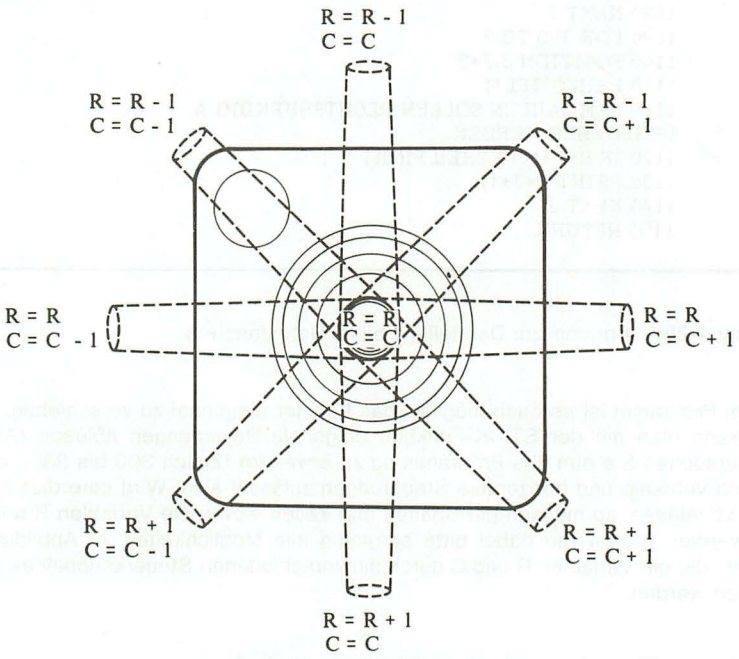
In diesem Programm ist es auch möglich, das Fenster diagnonal zu verschieben. Wie Sie wissen kann man mit der STICK-Funktion diagonale Bewegungen ablesen (Abbildung 4-26). Versuchen Sie nun das Programm so zu erweitern (Zeilen 300 bis 330), daß ausschließlich vertikale und horizontale Steuerungen zulässig sind. Wird eine diagonale Bewegung abgelesen, so müssen die Spalten und Zeilen, sowie die Variablen R und C verändert werden. Prüfen Sie dabei bitte sorgfältig alle Möglichkeiten. In Abbildung 4-30 sehen Sie, die die Variablen R und C durch die verschiedenen Steuerknüppelbewegungen manipuliert werden.

Numerische Eingaben mit dem Steuerknüppel

Nun werden Sie ein Programm kennenlernen, bei dem die numerischen Eingaben mit dem Steuerknüppel erledigt werden. Dabei wird zuerst eine Zahl dargestellt, die durch das Bewegen des Steuerknüppels verändert werden kann. Drückt man den Steuerknüppel nach links, so wird die Zahl kleiner. Drückt man ihn nach rechts, so wird sie größer. Steht der Knüppel wieder in der Normalposition, so verändert sich auch die Zahl nicht mehr. Wenn die Zahl, die Sie eingeben wollen, auf dem Schirm steht, dann drücken Sie den Feuerknopf. Geben Sie nun das Programm ein und probieren Sie es selbst aus:

```
1000 R=1
6509 REM LOESCHE VORHERIGEN WERT
6510 PRINT CHR$(125)
6519 REM STELLE JETZTIGEN WERT DAR
6530 PRINT R;
6539 REM MIT FEUERKNOPF PROGR. BEENDEN
6550 IF STRIG(0)=0 THEN END
6579 REM VORWAERTS?
6580 IF STICK(0)=11 AND R<>LO THEN R=R
-1:GOTO 6510
6589 REM RUECKWAERTS?
6590 IF STICK(0)=7 AND R<>HI THEN R=R+
1:GOTO 6510
6599 REM KEINE AENDERUNGEN
6600 GOTO 6550
```

Wie Sie sehen, läuft das Programm, aber es ist sehr schwierig, bei einer bestimmten Zahl anzuhalten. Das kommt daher, da das Programm die Stellung des Steuerknüppels sehr oft überprüft.



Anmerkung: R ist der Zeilen-Index und C der Spalten-Index.

Abbildung: 4-30: Auswirkung der Steuerknüppelbewegung auf die Variablen.

Man kann diesen Vorgang mit einer FOR-Next-Schleife verlangsamen.

```
6559 REM VERZOEGERUNGS-SCHLEIFE
6560 FOR J=1 TO 30:NEXT J
```

Diese Schleife gibt dem Anwender genug Zeit, die dargestellten Zahlen zu überprüfen und sie gegebenenfalls einzugeben. Eine größere Schleife würde unnötige Verzögerungen verursachen.
Das vollständige Programm sehen Sie in Abbildung 4-31. Dieses können Sie wiederum als Unterroutine einsetzen. Es bedient sich der gleichen Variablen wie die EINGABE-UNTERROUTINE von Abbildung 4-19. LO und HI spezifizieren die jeweils kleinsten, bzw. größten zulässigen numerischen Eingaben. Die Variablen IR und IC legen die Cursorpositionen fest und IL die Länge der Eingabe. Der eingegebene Wert wird der Variablen R zugewiesen. INC setzt die Stufen fest, in der die dargestellten Zahlen zunehmen, bzw. abnehmen (Zeilen 6580 und 6590). In diesem Programm wurden zwei verschiedene Geschwindigkeiten für die Verzögerungs-Schleife eingegeben (Zeile 6560). Am Anfang werden die Zahlen in sehr schneller Reihenfolge dargestellt. Sobald Sie aber den Steuer-

knüppel in die Normalstellung bringen, verändert sich die Geschwindigkeit (Zeile 6600). Wenn Sie den Steuerknüppel nur kurz nach rechts oder links drücken, werden Sie die verlangsamte Zahlendarstellung bemerken. Die hohe Geschwindigkeit erreicht man wieder durch längeres Drücken des Steuerknüppels.

```
10 DIM BL$(40)
19 REM FUELLE BL$ MIT LEERZEICHEN
20 BL$=" ":BL$(40)=BL$:BL$(2)=BL$
1000 PRINT CHR$(125)
1200 LO=0:HI=200:INC=0.5
1209 REM CURSOR POSITION
1210 IC=10:IR=11:IL=3
1220 GOSUB 6500
1230 PRINT
1240 PRINT "GEWUENSCHTE ZAHL: ";R
1900 END
6496 REM ****************************
6497 REM *NUMER.EINGABE M.STEUERKN. *
6498 REM ****************************
6499 REM LOESCHE EINGABEBEREICH
6500 POSITION IC,IR
6510 PRINT BL$(1,IL)
6519 REM STELLE JETZIGEN WERT DAR
6520 POSITION IC,IR
6530 PRINT R;
6539 REM MIT FEUERKNOPF PROGR. BEENDEN
6550 IF STRIG(0)=0 THEN RETURN
6559 REM VERZOEGERUNGS-SCHLEIFE
6560 FOR J=1 TO SD:NEXT J
6570 SD=1
6579 REM VORWAERTS?
6580 IF STICK(0)=11 AND R<>LO THEN R=R
-INC:GOTO 6510
6589 REM RUECKWAERTS?
6590 IF STICK(0)=7 AND R<>HI THEN R=R+
INC:GOTO 6510
6599 REM KEINE AENDERUNGEN
6600 SD=30:GOTO 6550
```

Abbildung: 4-31: Numerische Eingabe mit dem Steuerknüppel

Anwendung einer Auswahltabelle

Ihre Programme sollten immer so konzipiert sein, daß die Anzahl der Eingabemöglichkeiten für den Anwender so gering wie nur möglich gehalten wird. Dadurch werden von vornherein eventuelle Fehlerquellen beseitigt. Aber auch die Fragestellung spielt eine sehr wichtige Rolle. Haben Sie die Frage nicht korrekt formuliert, so ist es natürlich für den Anwender schwierig, die richtige Antwort einzugeben. In den Programmen, die wir bis jetzt behandelt haben, mußte der Anwender jeweils bestimmte Eingabefelder ausfüllen. Nun könnte der Computer anstatt "WAS MÖCHTEN SIE JETZT MACHEN?" auch die Frage stellen, "WELCHE AUSWAHL HABEN SIE GETROFFEN?". Und genau das macht das Menü (Auswahltabelle).
Erinnern Sie sich nun an das Programm von Abbildung 4-13; hier mußte ein Befehl einge-

geben werden. Die Wahlmöglichkeiten waren A, B, C, D, E oder H. Genau so kann auch ein Menü gestaltet werden (Abbildung 4-32). Unter Verwendung der Eingabe-Hilfsprogramme (Abbildung 4-16 bis 4-21), könnten Sie jetzt ein Programm schreiben, das ein Menü darstellt und anschließend Eingaben verarbeitet.
Die Verwendung eines Menüs hat für den Anwender, sowie für den Programmierer einige Vorteile. Der Anwender muß nicht erst lange überlegen, welche Eingabemöglichkeiten ihm zur Verfügung stehen. Der Programmierer dagegen, erspart sich viel Arbeit, weil er nun keine langen ausführlichen Eingabebeschreibungen mehr verfassen muß. Trotz alledem sollte das Programm jede Eingabe auf ihre Richtigkeit überprüfen, da nie gewährleistet ist, daß der Anwender immer eine der dargestellten Optionen wählt.

Auswahltabelle und Steuerknüppel

Der ATARI-Computer kann so programmiert werden, daß man den Cursor ausschließlich mit dem Steuerknüppel bewegen kann. Ist auf dem Bildschirm ein Menü dargestellt, so führt der Anwender den Cursor auf das von ihm ausgewählte Feld und drückt den Auslöseknopf.

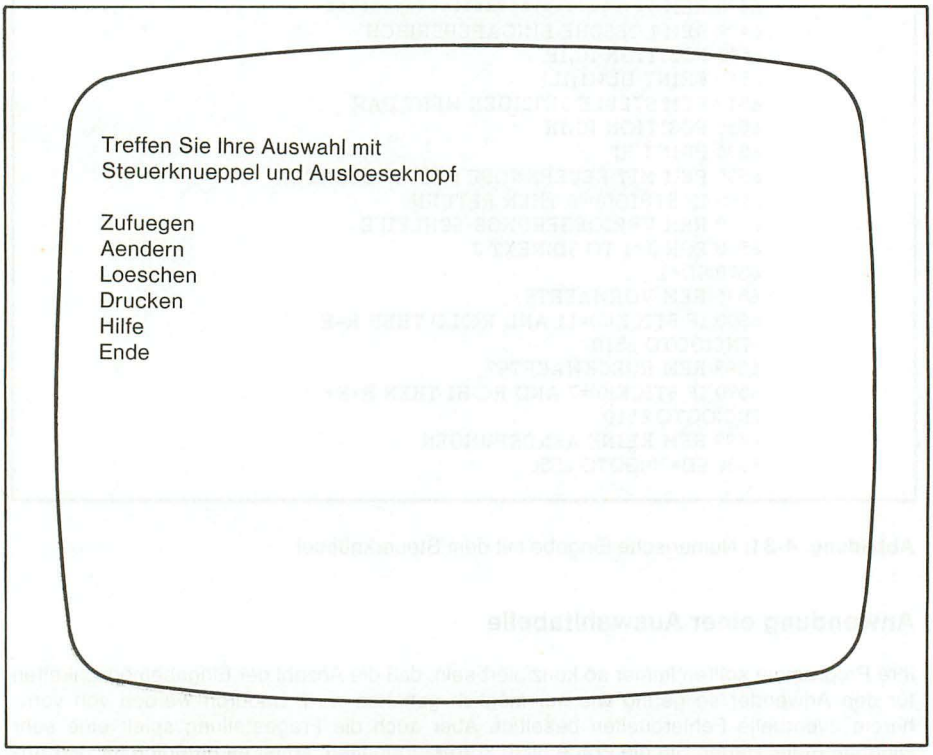


Abbildung 4-32: Darstellung einer Auswahltabelle

Das BASIC-Programm registriert das sofort und merkt sich außerdem, auf welcher Position der Cursor stand, als der Knopf gedrückt wurde.
Die Unterroutine von Abbildung 4-33 bewirkt, daß der Cursor mit Hilfe des Steuerknüppels

bewegt werden kann. In Zeile 6000 prüft das Programm zuerst, ob der Feuerknopf gedrückt wurde. Ist das der Fall, dann endet die Unterroutine hier und beläßt den Cursor auf seiner gegenwärtigen Position. Die Variable DLY1 setzt die Geschwindigkeit fest, in der die einzelnen Stuerknüppelstellungen abgelesen werden. Die FOR-NEXT-Schleife in Zeile 6010 dient somit ausschließlich zur Verzögerung. Dieses Unterprogramm findet also zuerst die Position des Cursors heraus, stellt die entsprechenden Cursorsteuerzeichen dar und aktualisiert die Positions-Variablen SC und SR (Zeilen 6020 bis 6060). Der Cursor hat jedoch nur einen begrenzten Bewegungsbereich, der durch folgenden Variable definiert wird:

.LC, die linke Spalten-Begrenzung
.RC, die rechte Spalten-Begrenzung
.TR, die obere Zeilen-Begrenzung
.BR, die untere Zeilen-Begrenzung

Geben Sie nun diese Programmzeilen zusammen mit der Unterroutine ein und lassen Sie es ablaufen:

```
999 REM LOESCHE BILDSCHIRM
1000 PRINT CHR$(125)
1010 POSITION 0,11
1020 PRINT "012345678901234567890123456
6789012345678 9"
1030 FOR J=0 TO 23
1040 POSITION 21,11
1050 PRINT J-INT(J/10)*10;
1060 NEXT J
1069 REM STARTPOSITION DES CURSORS
1070 POSITION 21,11
1080 PRINT CHR$(253);
1090 SC=21:SR=11
1099 REM FESTLEGUNG D. CURSORBEREICHES
1100 LC=2:RC=39:TR=0:BR=23
1109 REM FESTLEGUNG D.GESCHWINDIGKEIT
1110 DLY1=20
1120 GOSUB 6000
1900 END
```

Dieses Programm beginnt mit der Darstellung einer Zahlenreihe, mit deren Hilfe die Cursorbewegungen veranschaulicht werden sollen (Zeilen 1000 bis 1060)). Anschließend wird der Cursor auf die Mitte der Zeile gesetzt (Zeile 1070). Dann wird ein Klingelsignal ausgeführt (Zeile 1080). Bevor nun das Unterprogramm 'BEWEGEN DES CURSORS', aufgerufen wird, werden noch die später benötigten Variablen definiert.

```
5995 REM ****************************
5996 REM *    BEWEGEN DES CURSORS   *
5997 REM * MIT DEM STEUERKNUEPPEL   *
5998 REM ****************************
5999 REM M. FEUERKNOPF CURSOR STOPPEN
6000 IF STRIG(0)=0 THEN RETURN
6010 FOR J=1 TO DLY1:NEXT J
6019 REM PRUEFE STELLUNG D.KNUEPPELS
6020 IF STICK(0)=7 AND SC<>RC THEN SC=
SC+1:PRINT CHR$(31);
6030 IF STICK(0)=11 AND SC<>LC THEN SC
=SC-1:PRINT CHR$(30);
6040 IF STICK(0)=13 AND SR<>BR THEN SR
=SR+1:PRINT CHR$(29);
6050 IF STICK(0)=14 AND SR<>TR THEN SR
=SR-1:PRINT CHR$(28);
6060 GOTO 6000
```

Abbildung 4-33: Unterprogramm zum Bewegen des Cursors mit dem Steuerknüppel

Das folgende Programm läuft nur, wenn Sie auch die obenstehende Unterroutine eingeben. Achten Sie bitte darauf, daß sich das vorhergehende Programm nicht mehr im Speicher befindet.

```
999 REM LOESCHE BILDSCHIRM
1000 PRINT CHR$(125)
1010 PRINT "TREFFEN SIE IHRE AUSWAHL M
IT"
1015 PRINT "STEUERKNUEPPEL UND AUSLOES
EKNOPF"
1020 PRINT
1030 PRINT " ZUFUEGEN"
1040 PRINT " AENDERN"
1050 PRINT " LOESCHEN"
1060 PRINT " DRUCKEN"
1070 PRINT " HILFE"
1080 PRINT " ENDE"
1089 REM STARTPOSITION DES CURSORS=4,3
1090 POSITION 4,3
1100 PRINT CHR$(29);
1110 SC=4:SR=3
1119 REM FESTLEGUNG D. CURSORBEREICHES
1120 LC=4:RC=4:TR=3:BR=8
1129 REM CURSORGESCHWINDIGKEIT
1130 DLY1=30
1140 GOSUB 6000
1900 END
```

Zuerst stellt dieses Programm ein Menü dar (Zeilen 1000 bis 1080). Nun wird der Cursor auf dem ersten Buchstaben des obersten Befehlswortes plaziert (Zeilen 1090 bis 1100). Der Steuerknüppel kann den Cursor nur jeweils auf die ersten Buchstaben der Befehle setzen. Horizontale Bewegungen sind nicht möglich (Zeile 1120).
Wie Sie auch schon vom vorigen Programm wissen, legt man durch Drücken des Auslöseknopfes die Cursorposition fest. Diese Koordinaten enthalten nun die Variablen SC und SR.

Der Vollständigkeit halber, sollte das Programm noch die Cursorposition auswerten und den gewählten Befehl ausgeben. Natürlich würde ein richtiges Programm mehr tun, als nur das Wort darzustellen. Fügen Sie diese Zeilen in das letzte Programm ein:

```
1150 POSITION 2,0
1160 ON SR-2 GOSUB 2000,2500,3000,3500
,4000:GOTO 1900
1170 POSITION 2,10
1900 END
2000 PRINT "ZUFUEGEN":RETURN
2500 PRINT "AENDERN ":RETURN
3000 PRINT "LOESCHEN":RETURN
3500 PRINT "DRUCKEN ":RETURN
4000 PRINT "HILFE  ":RETURN
```

Ein- und Ausgabe von Daten im Einzelnen

Bis zu diesem Zeitpunkt haben wir nur mit solchen Anweisungen gearbeitet, die den Bildschirm automatisch als Ausgabegerät wählten. Die Befehle CSAVE und CLOAD dagegen, beziehen sich immer auf Kassettenrekorder. In den Programmen, die wir bis jetzt besprochen haben, wurde der INPUT-Befehl nur im Zusammenhang mit der Tastatur benutzt und der PRINT-Befehl sprach ausschließlich den Bildschirm an. Außer dem modifizierten INPUT und PRINT-Befehl, verfügt ATARI-BASIC noch über einige andere Ein/Ausgabe-Anweisungen. In diesem Kapitel werden Sie alle Ein/Ausgabe-Anweisungen kennenlernen, die die Tastatur, sowie den Bildschirm betreffen. In den nachfolgenden Kapitel werden die Eigenheiten der Befehle besprochen, die den Kassettenrekorder, den Drucker und das Diskettenlaufwerk betreffen.

Bezeichnung der Ein/Ausgabe-Einheit

Jedes Ein/Ausgabe-Gerät hat eine feststehende Bezeichnung. Der Bildschirm zum Beispiel, wird mit dem Kennzeichen S: angesprochen und die Tastatur mit K:.
In der direkten Betriebsart legt der ATARI-Computer den Bildschirm und das Tastenfeld zusammen. Als anzusprechende Einheit ergibt sich für ihn daraus, der Editor (Kennzeichen E:). Die weiteren Ein/Ausgabe-Einheiten und die dazugehörenden Kennzeichen werden später erklärt.

Ein/Ausgabe-Kanäle

Der Datenaustausch zwischen Ihrem ATARI-Computer und den angeschlossenen Peripheriegeräten erfolgt über Ein/Ausgabe-Kanäle. Hierfür stehen acht Kanäle zur Verfügung, die von 0 bis 7 durchnumeriert sind.
Bevor ein BASIC-Programm mit einem Ein/Ausgabe-Gerät kommunizieren kann, wird einer der Kanäle mit diesem Gerät verbunden. Auf diese Weise werden zum Beispiel die Ausgaben zu dem Kanal geleitet, der die Daten zu der vorher spezifizierten Ausgabe-Einheit leitet. Ein Programm arbeitet mit den Ein/Ausgabe-Kanälen genauso, wie Sie den Programmwähler bei Fernsehapparat benutzen. Man stellt das gewünschte Programm ein und empfängt somit die Sendung der Station, die man gewählt hat.
Wir benutzen in diesem Buch die Ausdrücke "Kanäle" oder Datenkanäle". In anderen Publikationen werden dafür auch folgende Begriffe verwendet: Ein/Ausgabe Steuerblock, IOCB oder File Nummer.
BASIC hat sich die Kanäle 0, 6 und 7 fest reserviert, wobei der Kanal Nummer 0 ausschließlich für den Editor vorgesehen ist. In der direkten Betriebsart werden alle Eingaben,

die auf dem Tasteneld eingetippt werden, über Kanal 0 weitergeleitet. Desgleichen benutzen auch die einfachen INPUT und PRINT-Befehle den Kanal 0. Kanal 7 ist für bestimmte Druckvorgänge, sowie für das Laden und Speichern von Programmen reserviert. Kanal 6 ist für die Anweisungen zum Erstellen von Bildschirm-Graphiken vorgesehen.
Die Datenkanäle 0 bis 5 dagegen, stehen ausschließlich für Ihr ATARI-BASIC Programm zur Verfügung. Unter bestimmten Voraussetzungen können Sie auch noch mit den Kanälen 6 und 7 arbeiten. Kanal 6 kann man nur benutzen, wenn das Programm keine Anweisungen zum Erstellen von Bildschirm-Graphiken enthält. Sind im Programm ferner keine Befehle zum Laden oder Speichern von Daten, sowie kein LPRINT-Befehl (wird in Kapitel 6 erklärt) vorhanden, dann steht Ihnen auch der Kanal 7 zur Verfügung.

Öffnen eines Datenkanals

Der OPEN-Befehl öffnet einen Datenkanal für die Ein- oder Ausgabe von Daten.
Dieser Befehl besteht aus dem Schlüsselwort OPEN, gefolgt von vier Parametern.
Zum Beispiel:

```
OPEN #1,12,0,"E:"
```

Die erste Zahl nach dem Wort OPEN bezeichnet die Nummer des Datenkanals. In unserem Beispiel wird der Kanal 1 für den Editor (E:) geöffnet.
Die zweite Nummer gibt die Kennzahl der Betriebsart an. Hierbei kann man zwischen verschiedenen Möglichkeiten wählen. Wichtig ist jedoch, daß die Logik stimmt. Man kann zum Beipiel nicht die Tastatur für die Ausgabe öffnen, da es unmöglich ist, Daten über die Tastatur auszugeben. Im allgemeinen spezifizieren diese Kennzahlen folgende Betriebsarten: 4 = Eingabe, 8 = Ausgabe und 12 = Ein- und Ausgabe. Die weiteren Betriebsarten werden zusammen mit den betreffenden Peripheriegeräten erklärt.
Der letzte Parameter vor dem Kennzeichen des angesprochenen Gerätes, ist eine zusätzliche Kennzahl, deren Bedeutung vom Peripheriegerät abhängig ist. Wird das Tastenfeld (K:) oder der Editor (E:) angesprochen, so wird diese Kennzahl vom Computer ignoriert. Wird jedoch der Bildschirm (S:) angesprochen, so kann mit dieser Nummer der Text- bzw. der Graphikmodus bestimmt werden. In diesem Kapitel beschäftigen wir uns nur mit dem Textmodus.
Öffnen Sie einen Datenkanal zum Editor (E:) oder zum Bildschirm im Textmodus (S:, dritter Parameter = 0), so wird zuerst der Bildschirm gelöscht. Probieren Sie dazu das folgende Programm aus.

```
10 DIM R$(1)
20 PRINT "DRUECKEN SIE 'RETURN' DAMIT
DER BEFEHL AUSGEFUEHRT WIRD"
25 INPUT R$
30 OPEN #1,4,0,"S:"
40 PRINT "DER BILDSCHIRM WURDE DURCH O
PEN GELOESCHT"
```

Schließen eines Datenkanals

Wurde ein Datenkanal geöffnet, so bleibt er solange offen, bis das Programm beendet ist oder der Befehl CLOSE eingegeben wird. Der Befehl END bewirkt, daß alle Kanäle geschlossen werden. Wird das Programm jedoch durch STOP, die BREAK-Taste oder einen Fehler unterbrochen, so bleiben alle geöffneten Kanäle offen.
Der CLOSE-Befehl hat folgende Form:

```
CLOSE #1
```

Der PRINT # Befehl

Nun kommen wir zu dem modifizierten PRINT-Befehl. Er ist mit dem bekannten PRINT-Befehl fast identisch, ermöglicht jedoch zusätzlich, die Daten direkt über einen geöffneten Kanal, also an andere Ausgabegeräte, weiterzuleiten. Der angesteuerte Datenkanal wird hierbei nach dem Schlüsselwort PRINT angegeben. Vergleichen Sie die beiden nachfolgenden Anweisungen:

```
10 PRINT #1;R$;R
20 PRINT R$;R
```

Die beiden Zeilen produzieren die gleiche Ausgabe. Zeile 20 leitet die Daten direkt an den Bildschirm weiter, während die erste Anweisung (Zeile 10) den Kanal 1 anspricht. Dieser muß natürlich vorher mit einem OPEN-Befehl geöffnet worden sein.
Bitte beachten Sie, daß wir nach der Nummer des Datenkanals (Zeile 10) ein Semikolon gesetzt haben. Verwendet man anstatt dessen ein Komma, so gibt der Computer Leerzeichen aus und beginnt mit der eigentlichen Ausgabe erst bei der nächsten Druckposition. Ein PRINT # Befehl, der nur eine Kanalnummer enthält, gibt ein EOL-Zeichen aus.
Wie Sie inzwischen wissen, bedeutet die Kennzahl 4, die Betriebsart "Eingabe". Beim Bildschirm verhält es sich damit jedoch anders. Egal ob Sie den Parameter 4 oder 8 angeben, der Bildschirm ist stets nur für die Ausgabe geöffnet. Sehen Sie dazu das folgende Beispiel:

```
10 OPEN #1,4,0,"S:"
20 PRINT #1,"OBWOHL DER DATENKANAL NUR
  FUER DIE EINGABE GEOEFFNET IST, WERDE
N DIE DATEN AUSGEGEBEN"
```

Der OPEN-Befehl in Zeile 10 löscht den Bildschirm und öffnet den Datenkanal 1 für die Eingabe. Wie Sie jedoch feststellen können, kümmert sich der Computer scheinbar nicht darum und gibt die Informationen trotzdem auf dem Bildschirm aus.

Der INPUT # Befehl

Der INPUT # Befehl beschränkt sich nicht nur auf Daten, die über die Tastatur eingegeben werden. Er kann vielmehr von allen Kanälen, die für die Eingabe geöffnet sind, Daten erhalten.
Dieser Befehl hat folgendes Format:

```
INPUT #1,R
```

Sie können nach der Kanalnummer entweder ein Komma oder ein Semikolon verwenden; das Resultat ist dasselbe.
Sobald Sie den INPUT # Befehl eingegeben haben, beginnt das Programm über den angegebenen Kanal, Daten einzulesen. Dieser Vorgang läuft solange, bis die RETURN-Taste gedrückt wird. Das dabei produzierte EOL-Zeichen, erkennt der Befehl als Datensatzende an. Anschließend versucht der INPUT # Befehl die eingelesenen Daten, der nächsten Variablen auf seiner Liste zuzuweisen. Ist das nicht möglich, so wird eine Fehlermeldung ausgegeben. Tippen Sie nun diese Zeilen ein:

```
10 OPEN #1,12,0,"E:"
20 INPUT #1,A
```

Haben Sie einen Unterschied zu dem normalen INPUT-Befehl festgestellt? Diese Form des Befehls gibt kein Fragezeichen auf dem Bildschirm aus. Das ist der einzige sichtbare Unterschied. Sie erinnern sich sicher an das Eingabe-Hilfsprogramm (Abbildung 4-19), das wir etwas früher in diesem Kapitel behandelt haben. Hier wurden die INPUT-Befehle benutzt, um Dateneingaben über die Tastatur zu ermöglichen. Vor jeder Eingabe-Maske wurde ein Fragezeichen dargestellt. Es gibt jedoch auch Fälle, in denen dieses Fragezeichen unpassend ist. Versuchen Sie nun, in diesem Unterprogramm die INPUT-Befehle durch INPUT # Befehle zu ersetzen. Im Hauptprogramm muß dazu natürlich erst ein Datenkanal für die Eingabe geöffnet werden.

Der PUT-Befehl

Der PUT-Befehl gibt einen einzelnen numerischen Wert über einen für die Ausgabe geöffneten Datenkanal aus. Ist der Kanal für den Bildschirm geöffnet, so wird dieser numerische Wert als ATASCII-Code interpretiert und das betreffende Zeichen wird auf dem Schirm dargestellt. Sehen Sie dazu das betreffende Programm:

```
10 OPEN #1,8,0,"E:"
20 PRINT "DER ATASCII CODE LAUTET";
30 INPUT R
40 PRINT "DAS ENTSPECHENDE ZEICHEN IST
: ";
50 PUT #1,R
60 PUT #1,155:REM EOL-ZEICHEN
69 REM M. BREAK TASTE PROGRAMM STOPPEN
70 GOTO 20
```

Genau wie der PRINT-Befehl, stellt auch PUT zuerst fest, an welcher Stelle die Daten dargestellt werden sollen. Diese Positionsangaben enthalten die Speicherplätze 84 (Zeile) und 85 (Spalte). Anders als PRINT, gibt PUT am Ende kein EOL-Zeichen aus. Deshalb muß dieses auch extra berücksichtigt werden (Zeile 60). Das Komma nach der Kanalnummer darf nicht durch ein Semikolon ersetzt werden, da das einen Fehler verursachen würde.

Der GET-Befehl

Der GET-Befehl liest ein einzelnes Zeichen über einen geöffneten Datenkanal ein. Das eingelesene bzw. eingegebene Zeichen wird nicht auf dem Bildschirm dargestellt. Drücken Sie nach der Eingabe des Zeichens nicht die RETURN-Taste.
Die eingelesenen Daten werden anschließend in ihre ATASCII-Codezahl umgewandelt. Geben Sie nun das Beispielprogramm ein und lassen Sie es ablaufen:

```
10 OPEN #1,4,0,"K:"
20 PRINT "DRUECKEN SIE EINE TASTE ";
30 GET #1,R
40 PRINT "SIE DRUECKTEN ";CHR$(R);",AT
ASCII CODE ";R
49 REM M. BREAK TASTE PROGRAMM STOPPEN
50 GOTO 20
```

Man kann das Programm auch so gestalten, daß es die Eingabe eines bestimmten Zeichens abwartet:

```
10 OPEN #1,4,0,"K:"
20 GET #1,R
30 IF R<>ASC("X") THEN GOTO 20
40 END
```

Dieses Programm prüft die eingegebenen Zeichen und wartet, bis das von ihm gewünschte 'X' eingetippt wird. Dann erst wird die Zeile 40 ausgeführt.
Der GET-Befehl wird häufig in Programmen eingesetzt, die einen Dialog mit dem Anwender führen. Die folgenden Zeilen zum Beispiel sind so geschrieben, daß der Anwender die RETURN-Taste drücken muß, damit das Programm weitergeht.

```
10 OPEN #1,4,0,"K:"
20 PRINT "SIND SIE SICHER?"
30 PRINT "DANN DRUECKEN SIE 'RETURN'."
40 GET #1,R
50 IF R<>155 THEN GOTO 40
60 PRINT "IN ORDNUNG, NUN MACHEN WIR W
EITER."
```

Bitte beachten Sie, daß eingetippte Zeichen nicht auf dem Bildschirm dargestellt werden.

Eingabe des Datums

In diesem Abschnitt werden wir ein Programm entwickeln, bei dem das laufende Datum eingegeben werden soll. Die Programmierung dieser Dateneingabe erfordert mehr Sorgfalt, als auf den ersten Blick notwendig scheint. Sehr wahrscheinlich wird das Datum nur eines unter vielen einzugebenden Daten sein. Deshalb ist es von größter Wichtigkeit, alle Dateneingaben sorgfältig vorzubereiten, um zu vermeiden, daß der Anwender bei einer fehlerhaften Eingabe alle Daten noch einmal wiederholen muß. Es sollen der Tag, der Monat und das Jahr in jeweils zweistelligen Ziffern eingegeben werden (Abbildung 4-34). Als Trennung zwischen den einzelnen Daten haben wir Striche benutzt. Man kann dafür natürlich auch andere Zeichen einsetzen.
Der Anwender sollte auf den ersten Blick erkennen können, wo die nächsten Daten einzugeben sind. Dabei wird uns wieder die schon bekannte Eingabe-Maske behilflich sein:

```
10 DIM MSK$(2),R$(2),DAT$(8),R(1)
20 MSK$="_"
1000 PRINT CHR$(125);"GEBEN SIE DAS DA
TUM EIN"
1010 IC=15:IR=4
1020 GOSUB 8200:REM EINGABE DES DATUMS
```

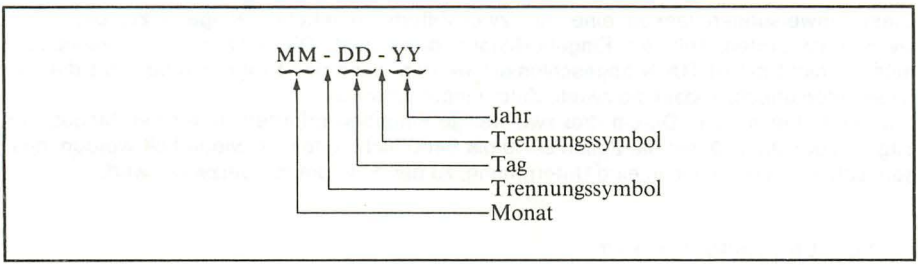


Abbildung 4-34: Format der Datum-Eingabe

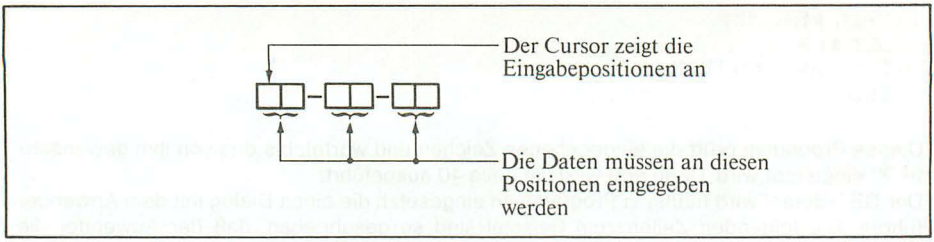


Abbildung 4-35: Eingabe-Maske für die Datum-Eingabe

```
1050 END
8200 POSITION IC,IR:PRINT CHR$(253);
8210 PRINT MSK$(1,2);"-";MSK$(1,2);"-"
;MSK$(1,2);
8220 POSITION IC,IR:PRINT CHR$(253);
8470 RETURN
```

In Programmzeile 1000 wird zuerst der Bildschirm gelöscht und anschließend um die Eingabe des Datums gebeten. Die erste Position der Eingabe-Maske ist in Zeile 4, Spalte 15 (Zeile 1010). Nach der Darstellung der Eingabe-Maske springt der Cursor wieder zurück zum ersten Zeichen der Maske.
Versuchen Sie nun, eine INPUT-Anweisung einzugeben, die den 'Tag' einlesen soll. Fügen Sie die folgende Zeile das Programm ein und lassen Sie es ablaufen:

```
8400 INPUT R$
```

Wie Sie sehen, geht es auf diese Weise nicht, da an der ersten Stelle der Eingabe-Maske ein Fragezeichen dargestelt wird. Ein INPUT # Befehl könnte hier Abhilfe schaffen, aber der Anwender hätte trotzdem noch die Möglichkeit, zu viele Zeichen einzugeben und somit die Maske zu ruinieren.
Dies ist eine gute Gelegenheit, den GET-Befehl anzuwenden. Geben Sie die folgenden Programmzeilen ein:

```
59 REM OEFFNE TASTATUR ZUR EINGABE
60 OPEN #1,4,0,"K:"
8400 FOR J1=0 TO 1
8410 GET #1,R
8440 PRINT CHR$(R);
8450 NEXT J1
```

Diese Anweisungen lassen eine aus zwei Ziffern bestehende Eingabe zu. Die Daten werden im ersten Teil der Eingabe-Maske dargestellt. Die einzelnen Dateneingaben müssen nicht mit RETURN abgeschlossen werden, denn das Programm beendet die Eingabe automatisch, sobald die zweite Ziffer eingetippt wird.
Nun sind aber für das Datum drei zweistellige Eingaben erforderlich: für den Monat, den Tag und das Jahr. Damit die Zeilen 8400 bis 8460 nicht jedesmal wiederholt werden müssen, schreiben wir diese in eine Unterroutine, zu der dann dreimal verzweigt wird.

```
8230 GOSUB 8400:REM MONAT
8270 PRINT CHR$(31);
8280 GOSUB 8400:REM TAG
```

```
8340 PRINT CHR$(31);
8350 GOSUB 8400:REM JAHR
8380 RETURN
8397 REM ****************************
8398 REM *  EINGABE VON ZWEI ZIFFERN*
8399 REM ****************************
8400 FOR J1=0 TO 1
8410 GET #1,R:R(J1)=R
8440 PRINT CHR$(R);
8450 NEXT J1
8460 R$=CHR$(R(0)):R$(2)=CHR$(R(1))
8470 RETURN
```

Dem Anwender werden drei Hilfestellungen gegeben, um Eingabefehler zu vermeiden, bzw. zu korrigieren:

- Das Programm akzeptiert nur numerische Eingaben
- Es überprüft, ob die Monats-, Tages- oder Jahresangaben innerhalb eines festgelegten Gültigkeitsbereiches liegen
- Ermöglichen Sie dem Anwender, die Dateneingaben erneut zu starten.

Abbildung 4-36 zeigt den kompletten Ausdruck des Programms zur Datum-Eingabe. Das Programm läßt nur numerische Eingaben zu (Zeile 8430). Die Monatsangabe muß eine Zahl zwischen 1 und 12 sein (Zeile 8250). Schaltjahre bleiben hierbei unberücksichtigt, jedoch wird die Tagesangabe in Bezug auf den Monat überprüft (Zeilen 8290 bis 8320). Das Programm verarbeitet Jahresangaben von 0 bis 99 (Zeile 8360). Werden Daten eingegeben, die außerhalb der jeweiligen Ober- bzw. Untergenze liegen, so müssen die gesamten Eingaben wiederholt werden. Drückt der Anwender die BACK S-Taste, so wird ebenso die vollständige Eingabe wiederholt (Zeile 8420).
Beachten Sie bitte, daß die Monats-, Tages- oder Jahresdaten in dem achtstelligen String DAT$ abgelegt werden (Zeilen 8260, 8330 und 8370).
Wie Sie sehen, kostet es mehr Zeit, ein Eingabeprogramm zu schreiben, das die Informationen in einer ansprechenden Form darstellt, die Richtigkeit der Eingaben überprüft und dem Anwender die Korrektur der Daten jederzeit ermöglicht. Diese Mehrarbeit lohnt sich jedoch auf jeden Fall, da Sie das Programm nur einmal schreiben müssen. Der Anwender dagegen, benutzt das Programm viele hundert Male. Investieren Sie daher zusätzliche Programmierzeit, um dem Anwender unnötigen Ärger und Verzögerungen bei den Programmbenutzungen zu ersparen.

Eine String-Eingabe-Unterroutine

Die allgemeine Eingabe-Unterroutine (Abbildung 4-19), die wir am Anfang des Kapitels entwickelt haben, weist noch einige Unzulänglichkeiten auf. Sicherlich haben Sie bemerkt, daß der Anwender hierbei Eingaben tätigen kann, die länger sind, als die festgelegte Eingabe-Maske. Noch schlimmer ist jedoch, daß man den Cursor mit den Steuerungstasten über den ganzen Bildschirm bewegen kann. Das führt früher oder später unweigerlich dazu, daß der Anwender die gesamte Darstellung durcheinander bringt. Tauscht man jedoch den INPUT-Befehl durch den GET-Befehl aus, so hat man viel mehr Möglichkeiten, die Eingaben zu kontrollieren und zu begrenzen. Außerdem kann man sich auf diese Weise der lästigen Fragezeichen erledigen, die vor jeder Dateneingabe auftauchen.
In Abbildung 4-37 sehen Sie ein Eingabe-Unterroutine in ihrem "rohen" Zusand. Zuerst wird die Eingabe-Variable geleert, um sicherzustellen, daß alle eventuell darin abgelegten, alten Werte gelöscht sind (Zeile 600). Um die Bildschirmgestaltung etwas übersichtlicher zu formen, wird nun noch der Cursor abgeschaltet (Zeile 610). Die Zeilen 630 und 640 veranlassen nun die Darstellung der Eingabe-Maske. Im Hauptprogramm muß zuvor die Masken-Variable MSK$ dimensioniert und definiert werden. Desgleichen müssen folgenden Variablen Werte zugewiesen werden: IC (Spaltenposition), IR (Zeilenposition) und IL

```
10 DIM MSK$(2),R$(2),DAT$(8),R(1)
20 MSK$="_"
59 REM OEFFNE TASTATUR ZUR EINGABE
60 OPEN #1,4,0,"K:"
1000 PRINT CHR$(125);"GEBEN SIE DAS DA
TUM EIN"
1010 IC=15:IR=4
1020 GOSUB 8200:REM EINGABE D.DATUMS
1030 POSITION 15,6
1040 PRINT DAT$
1050 END
8197 REM ******************************
8198 REM *   EINGABE DES DATUMS    *
8199 REM ******************************
8200 POSITION IC,IR:PRINT CHR$(253);
8210 PRINT MSK$(1,2);"-";MSK$(1,2);"-"
;MSK$(1,2);
8220 POSITION IC,IR:PRINT CHR$(253);
8230 GOSUB 8400:REM MONAT
8240 M=VAL(R$)
8250 IF M<1 OR M>12 THEN 8200
8260 DAT $(1,2)=R$:DAT$(3,3)="-"
8270 PRINT CHR$(31);
8280 GOSUB 8400:REM TAG
8290 T=VAL(R$)
8300 IF T<1 THEN 8200
8310 IF M=2 AND T>29 THEN 8200
8320 IF (M=4 OR M=6 OR M=9 OR M=11) AN
D T>30 THEN 8200
8325 IF T>31 THEN 8200
8330 DAT$(4,5)=R$:DAT$(6,6)="-"
8340 PRINT CHR$(31);
8350 GOSUB 8400:REM JAHR
8360 IF J<0 OR J>99 THEN 8200
8370 DAT$(7,8)=R$
8380 RETURN
8397 REM ******************************
8398 REM *  EINGABE VON ZWEI ZIFFERN*
8399 REM ******************************
8400 FOR J1=0 TO 1
8410 GET #1,R:R(J1)=R
8419 REM NEUSTART MIT 'BACK S' TASTE
8420 IF R=126 THEN POP:GOTO 8200
8430 IF R<48 OR R>57 THEN 8410
8440 PRINT CHR$(R);
8450 NEXT J1
8460 R$=CHR$(R(0)):R$(2)=CHR$(R(1))
8470 RETURN
```

Abbildung 4-36: Programm zur Datum Eingabe

(maximale Länge der Eingabe). Mit dem GET-Befehl in Zeile 660, liest die Eingabe-Unterroutine einen einzelnen Wert ein. Dabei wird vorausgesetzt, daß im Hauptprogramm der Kanal 1 für die Eingabe über die Tastatur geöffnet wurde. Nun wird die gegenwärtige Länge der Dateneingabe berechnet (Zeile 680). Die Ziffern, Großbuchstaben und Interpunktionszeichen werden solange an den Eingabe-String gehängt, bis seine festgelegte Länge erreicht ist (Zeile 690). Drückt der Anwender jedoch die BACK S-Taste, so wird das zuletzt eingetippte Zeichen gelöscht und die Eingabe-Maske erneut dargestellt (Zeilen 710 bis 750).
Es gibt natürlich noch viele Möglichkeiten, wie man die String-Eingabe-Unterroutine verbessern könnte. Dazu einige Anregungen:

- Setzen Sie Variable ein, um den Eingabe-Kanal zu spezifizieren.
- Ermöglichen Sie die Eingabe von Groß- und Kleinbuchstaben (ATASCII-Code 97 bis 122 für Kleinbuchstaben).
- Schreiben Sie eine Unterroutine, die erweiterte Eingabe-Informationen darstellt, sobald ein bestimmtes Zeichen eingetippt wird. Die Programmierung dieses Vorganges ist natürlich etwas schwieriger, als in der Eingabe-Unterroutine von Abbildung 4-19, da nun jedes Zeichen durch das Unterprogramm überprüft werden muß.
- Das Programm sollte numerische Eingaben nur innerhalb eines festgelegten Zahlenbereiches zulassen. Überprüfen Sie die Zahlengröße mit: (LO<=R<=HI). In der letzten Stufe, bevor das prüfende Unterprogramm wieder zurückverzweigt, sollte kontrolliert werden, ob die Eingabe ein numerischer Wert sein muß. Ist das der Fall, so muß die String-Eingabe in einen numerischen Wert umgewandelt und nachgerechnet werden, ob diese Zahl innerhalb des festgelegten Zahlenbereiches liegt. Benutzen Sie dafür den TRAP-Befehl und die Unterroutine zur Fehlerbehandlung, die sich mit nicht-numerischen Eingaben befassen.

Abbildung 4-37 zeigt Ihnen nur die String-Eingabe-Unterroutine selbst. Dieses Programm läuft nur, wenn sich die folgenden Zeilen bereits im Speicher befinden.

```
9 REM VARIABLEN WERDEN INITIALISIERT
10 DIM MSK$(40),R$(20)
30 MSK$="_":MSK$(40)=MSK$:MSK$(2)=MSK$
49 REM OEFFNE TASTATUR ZUR EINGABE
50 OPEN #1,4,0,"K:"
89 REM VERZWEIGEN ZUM HAUPTPROGRAMM
90 GOTO 1000
1000 PRINT CHR$(125)
1010 POSITION 2,4
1020 PRINT "NAME"
1030 IC=7:IR=4:IL=20:GOSUB 600
1040 POSITION 2,8
1050 PRINT "EINGEGEBEN WURDE: ":R$
1060 CLOSE #1
1070 END
```

Außerkraftsetzen der BREAK-Taste

Selbst ein sorgfältig konzipiertes Programm kann Schwachstellen haben. Eine davon ist die BREAK-Taste, durch die der Anwender den Programmablauf jederzeit stoppen kann. Nach Betätigung dieser Taste gibt der Computer eine Meldung aus, die den ganzen Bildschirmaufbau zerstört. In fast allen Fällen kann das Programm nun auch nicht mehr mit CONT weitergeführt werden, da die Programmausführung erst wieder in der folgenden Zeile einsetzen würde.
Da die BREAK-Taste wie schon bekannt, eine Zeile auch am Anfang oder in der Mitte unterbrechen kann, werden die restlichen Befehle dieser Programmzeile nicht mehr ausgeführt.

```
597 REM ******************************
598 REM *STRING-EINGABE-UNTERROUTINE *
599 REM ******************************
600 R$=""
610 POKE 755,0:REM CURSOR VERSCHWINDET
629 REM DARSTELLUNG DER EINGABE-MASKE
630 POSITION IC,IR
640 PRINT MSK$(1,IL)
650 POSITION IC,IR
660 GET #1,R
670 J=LEN(R$)
679 REM NACH DRUECKEN DER RETURN TASTE
WIRD DER CURSOR WIEDER SICHTBAR
680 IF R=155 THEN POKE 755,2:RETURN
689 REM UEBERPRUEFE JEDES ZEICHEN
690 IF R>=32 AND R<=95 AND J<IL THEN R
$(J+1,J+1)=CHR$(R):PRINT CHR$(R);:GOTO
 660
700 IF R<>126 OR J=0 THEN GOTO 660

709 REM ERNEUTE DARSTELLUNG D. EINGABE
-MASKE U. LOESCHEN D. LETZTEN ZEICHENS
710 POSITION IC+J-1,IR
720 PRINT MSK$(1,1);
730 POSITION IC+J-1,IR
740 IF J>1 THEN R$=R$(1,J-1)
750 IF J=1 THEN R$=""
760 GOTO 660
```

Abbildung 4-37: Programmausdruck der String-Eingabe-Unterroutine

Es gibt jedoch eine Möglichkeit, die Funktion der BREAK-Taste vorübergehend außer Kraft zu setzen (Abbildung 4-38). Leider bietet auch das keine hunderprozentige Sicherheit, da verschiedene andere Befehle bzw. Tasten, die BREAK-Taste wieder reaktivieren können. Wie zum Beispiel: die SYSTEM RESET-Taste – die erste PRINT-Anweisung, die ausgeführt wird – jeder OPEN-Befehl, der einen Datenkanal zum Bildschirm öffnet (Kennzeichen E: oder S:) – die erste PRINT-Anweisung nach einem derartigen OPEN-Befehl – sowie die GRAPHICS-Anweisungen (siehe Kapitel 8). Der einfachste Weg, diese Reaktivierung zu umgehen ist, immer wieder zu der Unterroutine zu verzweigen, die die BREAK-Taste außer Kraft setzt (Abbildung 4-38). Von dieser Möglichkeit sollten Sie vor allem in Eingabe-Unterprogrammen Gebrauch machen. In der Eingabe-Unterroutine (Abbildung 4-19), sowie in der String-Eingabe-Unterroutine (Abbildung 4-37), könnte man diesen Programmteil gut einbauen. Zum Beispiel mit:620 GOSUB 850.

LOCATE

Nun kommen wir zu einen neuen Befehl von ATARI-BASIC. Der LOCATE-Befehl liest die ATASCII-Code-Nummer eines bestimmten Zeichens vom Bildschirm. Er hat folgendes Format:

```
LOCATE 3,4,AC
```

Der Befehl enthält die horizontale Position des zu lesenden Zeichens an erster Stelle und die vertikale Position an zweiter Stelle. Anschließend wird die Variable (in unserem Beispiel AC) angegeben, in der der ATASCII-Code des gelesenen Zeichens abgelegt wird.

Bevor Sie den LOCATE-Befehl einsetzen können, muß der Datenkanal 6 für die Eingabe vom Bildschirm geöffnet sein.
Dieser Befehl liest nicht nur Zeichen vom Schirm, sondern bewegt den Cursor gleichzeitig auch um jeweils eine Stelle nach rechts. Am Zeilenende bewirkt diese Funktion, daß der Cursor an den Anfang der nächsten Zeile springt. Genau wie beim POSITION-Befehl, enthält auch hier der Speicherplatz 84 die Zeile und Platz 85 die Reihe. Will man nun bei LOCATE verhindern, daß sich der Cursor automatisch nach rechts bewegt, so muß man dafür sorgen, daß die Speicherplätze 84 und 85 nicht mit der gegenwärtigen Cursorposition aktualisiert werden. Sehen Sie dazu die untenstehende Programmzeilen:

```
1000 OPEN #6,12,0,"S:"
1400 P84=PEEK(84):P85=PEEK(85)
1410 LOCATE C,R,CODE
1420 POKE 84,P84:POKE 85,P85
```

Zeichen-Eingabe mit dem Steuerknüppel

Etwa in der Mitte dieses Kapitels haben wir uns damit beschäftigt, wie man numerische Werte allein mit dem Steuerknüppel eingeben kann. Mit Hilfe des LOCATE-Befehls und des Steuerknüppels, kann das Programm jedes Zeichen direkt vom Bildschirm einlesen. Zuerst wird eine Reihe mit unterschiedlichen Zeichen dargestellt, aus denen der Anwender auswählen kann.

```
846 REM ******************************
847 REM *     AUSSERKRAFTSETZEN   *
848 REM *       DER BREAK TASTE   *
849 REM ******************************
850 J=PEEK(128)-128
860 IF J<0 THEN RETURN
870 POKE 16,J
880 POKE 53774,J
890 RETURN
```

Abbildung 4-38: Unterprogramm zum Außerkraftsetzen der BREAK-Taste

Der Cursor wird mit Hilfe des Steuerknüppels von Zeichen zu Zeichen bewegt, bis er auf dem gewünschten Symbol steht. Drückt der Anwender nun den Auslöseknopf, so liest der LOCATE-Befehl den Wert vom Bildschirm direkt in das Programm ein. Das nächste Programm veranschaulicht Ihnen diesen Vorgang. Geben Sie vorher das Unterprogramm von Abbildung 4-33 ein.

```
19 REM ATASCII CODE DER ZEICHEN
20 DATA 0,16,19,20,96,123
999 REM OEFFNE EIN/AUSGABEKAN.F.SCHIRM
1000 OPEN #6,12,0,"S:"
1010 PRINT CHR$(125)
1019 REM DARSTELLUNG DER ZEICHEN
1020 PRINT "WAEHLEN SIE EIN ZEICHEN"
1030 RESTORE 20
1040 FOR J=3 TO 8
```

```
1050 READ ZEICHEN
1060 POSITION 4,J
1070 PRINT CHR$(ZEICHEN)
1080 NEXT J
1089 REM CURSORPOSITION IST 4,3
1090 POSITION 4,2
1100 PRINT CHR$(29);
1110 SC=4:SR=3
1119 REM FESTLEGUNG D.CURSORGEBIETES
1120 LC=4:RC=4:TR=3:BR=8
1129 REM CURSORGESCHWINDIGKEIT
1130 DLY1=30
1140 GOSUB 6000
1149 REM ZEICHEN LESEN
1150 LOCATE SC,SR,ZEICHEN
1160 ZEICHEN=ZEICHEN-128*SGN(ZEICHEN-1
27)
1170 POSITION 2,12
1180 PRINT "SIE WAEHLTEN ";CHR$(ZEICHE
N)
1900 END
```

Wie Sie sehen, stellt das Programm zuerst sechs Symbole dar (Zeile 20 und Zeilen 1000 bis 1070). Anschließend wird der Cursor auf das erste Zeichen gesetzt (Zeilen 1090 und 1100), das Gebiet festgelegt, in dem man den Cursor verschieben kann (Zeile 1120) und die Geschwindigkeit, in der der Cursor durch den Steuerknüppel bewegt wird (Zeile 1130). Der Anwender hat somit die Möglichkeit, den Cursor auf das Zeichen zu führen, das er wählen will (Zeile 1140). Nun liest der LOCATE-Befehl den ATASCII-Code des Zeichens unter dem Cursor (Zeile 1150). Da sich der Cursor in dem Moment, in dem der Wert abgelesen wird auf dem Zeichen steht, wird die ATASCII-Code-Zahl des invertierten Symbols ermittelt. In Zeile 1160 wird zum Schluß daraus wieder das aktuelle (nicht-invertierte) Zeichen berechnet.

Fehlersuche in Programmen

Ein neues Programm arbeitet fast nie so, wie Sie sich das eigentlich vorgestellt haben. Ist die BASIC-Syntax korrekt, dann haben sich wahrscheinlich in der Programmlogik Fehler versteckt. Diese Programmfehler zu finden und zu eliminieren, ist das, was man Fehlersuche in Programmen nennt.
Vorher noch ein wichtiger Hinweis: Lassen Sie sich beim Programmieren genügend Zeit, planen Sie Ihre Schritte sorgfältig und versuchen Sie gleich zu Anfang, Fehler zu vermeiden. Fangen Sie nicht an, einfach draufloszutippen, solange Sie sich über das Programm, das Sie schreiben möchten, nicht vollständig im Klaren sind. Wenn Sie erst mit dem Programmieren anfangen, dann sollten Sie die in Anhang I aufgeführte Literatur ebenfalls zu Rate ziehen.
Es mag Sie jetzt vielleicht überraschen, aber die PRINT-Anweisung erweist sich bei der Fehlersuche als wertvolles Werkzeug. Sie können zum Beispiel einige zusätzliche PRINT-Befehle an strategischen Punkten in Ihr Programm einbauen, die Meldungen darstellen, wenn das Programm diese Punkte ohne Fehler passiert hat. Das erleichtert Ihnen das Überprüfen des Programmablaufes. Außerdem könnten zusätzliche PRINT-Anweisungen den gegenwärtigen Wert bestimmter Variabler ausgeben. Dadurch können Sie mehr Informationen über den Programmstand erhalten. Ferner kann man auf diese Weise Fehler in komplizierten Berechnungen aufspüren.

Optimales Programmieren

Ein optimales Programm sollte eine sehr kurze Ausführungszeit und sehr wenig Speicherkapazität benötigen. Einen besseren Maßstab für die Vorzüge eines Programms bietet jedoch seine Verwendbarkeit und Praxisnähe. Was nützt letzten Endes ein raffiniertes Programm, wenn der Anwender nichts damit anfangen kann und das eigentliche Ziel des Programms, nämlich eine Tätigkeit zu erleichtern, darüber vergessen wurde. Natürlich können auch praktische Programme raffiniert konzipiert sein. Ein schnelles Programm erfordert vom Anwender weniger Ausdauer und Geduld als ein langsames. Gehen Sie mit der Speicherkapazität sparsam um, so bleibt noch Platz, Funktionen einzubauen, die die Programmbenutzung erleichtern können. Anschließend werden wir einige Wege aufzeigen, wie man schnellere Programme schreibt und wenig Speicherkapazität benötigt.
Einige dieser Programmiertechniken die helfen, Ihr Programm zu beschleunigen, nehmen dafür freilich mehr Speicherkapazität in Anspruch. Andere, die weniger Kapazität benötigen, laufen dafür langsamer. In diesen Fällen müssen Sie sich entscheiden, was für Ihr spezielles Programm wichtiger ist.

Schnelle Programme

Bevor Sie die erste Anweisung Ihres Programms schreiben, sollten Sie den ganzen Ablauf sorgfältig durchdacht und geplant haben. Dazu folgende Ratschläge:

- Fassen Sie die Abschnitte zusammen, die in Ihrem Programm die meiste Zeit beanspruchen: Initialisierung der Datenfelder und Strings, Längen-Berechnungen, Bildschirmaufbau usw. Nun überlegen Sie sich Wege, wie diese Programmteile am schnellsten umgesetzt werden können.

- Die Unterroutinen, die vom Programm häufig aufgerufen werden, sollten mit niedrigen Zeilennummern beginnen. Ebenso verfahren Sie mit FOR-NEXT-Schleifen, die wiederholt benutzt werden, denn BASIC beginnt die Suche nach einer Zeilennummer immer am Anfang des Programms. Deshalb findet es die niedrigeren Zeilennummer natürlich auch schneller als die hohen.

- Wenn Sie mit verschachtelten FOR-NEXT-Schleifen arbeiten, dann sollten diese so konstruiert sein, daß die Schleife mit den meisten Wiederholungen ganz innen liegt. Dadurch kann das Hin- und Herspringen zwischen den Schleifen auf ein Minimum reduziert werden.

- Anstatt Berechnungen zu wiederholen, sollten Sie diese nur einmal durchführen, den Wert einer Variablen zuweisen und diese Variable entsprechend oft einsetzen.

- Wenn möglich, sollten Sie mit einfachen Berechnungen arbeiten. Für Multiplikationen und Divisionen braucht der Computer länger, als für Additionen und Subtraktionen. Am meisten Zeit nimmt die Bildung von Potenzen in Anspruch. Ebenso dauern arithmetische Funktionen, insbesondere verschachtelte Funktionen, ziemlich lange. Innerhalb einer FOR-NEXT-Schleife sollten Sie möglichst keine Berechnungen verwenden. Anstatt dessen lassen sich Index Variablen und Zählschritte (STEP) geschickt einsetzen.

Sobald ihr Programm läuft, sollten Sie noch einmal zum Anfang gehen und es gegebenenfalls umschreiben. Während der Fehlersuche haben Sie wahrscheinlich einige neue Variablen eingefügt. Diese sollten Sie jetzt zusammenlegen und wenn möglich, durch bereits im Programm vorhandene, ersetzen. Auf diese Weise gestalten Sie Ihr Programm übersichtlicher und erleichtern sich spätere Veränderungen.

Kompakte Programme

Sobald Sie anfangen, ihr Programm zu entwerfen, müssen Sie auch daran denken, Speicherkapazität zu sparen. Verwenden Sie dazu die untenstehenden Anregungen. Aber mit Vorsicht, denn einige von Ihnen führen dazu, daß die Programme anschließend schwierig zu entziffern sind. Wie Sie aus Erfahrung wissen, ist es schon mühsam genug, beim ersten Mal herauszufinden, wie man das Programm zum Laufen bringen kann. Noch schwieriger wird es jedoch, wenn Sie die Abläufe nach einiger Zeit vergessen haben und das ganze Programm vor jeder Benutzung nocheinmal durcharbeiten müssen.

- Verwenden Sie nach Möglichkeit keine Konstanten (e. g. 0, 100, "Y"). Anstatt dessen weisen Sie einer Variablen den Wert der Konstanten zu. Nun setzen Sie diese Variable in der Art und Weise ein, wie Sie das auch mit der Konstanten gemacht hätten. Ein wesentlicher Vorteil dabei ist, daß man einer Variablen problemlos andere Werte zuweisen kann, während das Austauschen von Konstanten sehr viel umständlicher ist.

- Benutzen Sie Unterroutinen, in denen man sich wiederholende Programmzeilen unterbringen kann. Das hat den Vorteil, daß Ihr Programm besser zu lesen, zu verändern und nachzuvollziehen ist. Außerdem wird auf diese Weise Speicherkapazität gespart.

- Verwenden Sie auch das Null-Element der Datenfelder (zum Beispiel: X(0), B(0)).

- Initialisieren Sie die Variablen mit READ und DATA, nicht mit den einfachen LET-Anweisungen. Noch besser wäre, wenn Sie INPUT, GET und Dateien (siehe Kapitel 5 und 7) dafür benutzen würden.

- Ersetzen Sie beim Verzweigen zu Programmabschnitten die konstanten Zeilennummern durch Variable.

- Haushalten Sie mit den Variablen sparsam. Es erweist sich als vorteilhaft, wenn man Standard-Variablen als Index für FOR-NEXT-Schleifen, für Zwischenberechnungen usw. einsetzt und diese unverändert wiederbenutzt. Natürlich dürfen Sie dabei nicht den Überblick verlieren. Jedoch können einheitliche Variablen dazu beitragen, daß das Programm übersichtlicher wird (zum Beispiel, enthält R$ immer die Antwort des Anwenders).

- Schreiben Sie mehrere Anweisungen in eine Programmzeile, aber denken Sie auch daran, daß zusammengefügte Zeilen schwerer zu lesen und zu verstehen sind.

- Fügen Sie die REM-Anweisungen häufig in Ihre Programme ein. Je mehr Sie erklärende Bemerkungen einflechten, um so schneller haben Sie wieder den Überblick über den Programmablauf. Diese Anmerkungen beweisen sich als besonders hilfreich, wenn man sich längere Zeit nicht mehr mit dem Programm beschäftigt hat.

Schreiben Sie nun Ihr Programm um und beachten Sie alle Ratschläge. Dadurch wird das Programm nicht nur schneller, sondern Sie sparen auch noch wertvollen Speicherplatz ein.

Programmieren in Maschinensprache

In der Form, in der wir die BASIC-Befehle eingeben, kann sie der ATARI-Computer nicht verstehen. Er muß sie dazu in eine andere Sprache, und zwar in die Maschinensprache umsetzen. Diese besteht nicht aus Worten, wie PRINT oder OPEN, sondern ausschließlich aus Zahlen. Jeder Befehl dieser Sprache hat ebenfalls einen Namen, aber der Computer

benutzt nur die numerischen Bezeichnungen. Außerdem werden in der Maschinensprache nur Konstante und keine Variable verwendet.
Das Programmieren in der Maschinensprache ist viel komplizierter, als in BASIC. Wo liegt also der Sinn? Nun, mit der Maschinensprache haben Sie eine vielfach größere Kontrolle über die Aktivitäten Ihres Computers. In etwa läßt sich das mit dem Einsatz von PEEK und POKE vergleichen und der daraus gewonnenen Kontrolle. Jedoch kennt die Maschinensprache noch zahlreiche andere wirkungsvolle Möglichkeiten.
Es gibt eine Computersprache, die der Maschinensprache ziemlich ähnlich ist: die Assembler-Sprache. Diese Sprache setzt sich mnemonischen Befehlen zusammen; das sind die Namen der numerischen Befehle aus der Maschinensprache. Die Anweisungen in Assembler stimmen demnach mit den jeweiligen Anweisungen der Maschinensprache überein. In den meisten Fällen übernimmt der Computer die Umwandlung der in Assembler geschriebenen Programme, in die Maschinensprache. Dessen ungeachtet, kann man auch direkt in Maschinensprache programmieren. Hierbei ist zu beachten, daß es mehrere derartige Sprachen gibt. Ihr ATARI-Computer versteht eine Maschinensprache.
Dieses Buch beschäftigt sich nicht mit dem Programmieren in Assembler oder in der Maschinensprache. Wenn Sie mehr über diese Themen wissen möchten, dann sehen Sie in Anhang I nach, wo Sie Hinweise zu weiterer Literatur finden.

Die USR-Funktion

In ATARI BASIC haben Sie die Möglichkeit, ein in Maschinensprache geschriebenes Programm aufzurufen und anschließend wieder zum BASIC-Programm zurückzukehren. Dazu wird die USR-Funktion eingesetzt. Hier ein Beispiel:

```
1500 A=USR(1664)
```

Man kann diese Funktion auch als Variable oder als Ausdruck verwenden. Sie ergibt in jedem Fall einen numerischen Wert.
In der Klammer hinter USR muß mindestens ein Parameter angegeben sein. Dieser definiert die Startadresse, ab der das Maschinenprogramm ausgeführt werden soll. Die weiteren Parameter werden durch Kommas voneinander getrennt. Ihre Werte müssen zwischen 0 und 65535 liegen. BASIC überträgt den Wert der Parameter über den Hardware Stack (Stapelspeicher).
Das folgende Beispiel enthält die USR-Funktion mit vier Parametern:

```
1550 PRINT USR(MLA,2,RND(0)*255,100)
```

Der Hardware Stack

Sobald ATARI-BASIC auf eine USR-Funktion stößt, wird die gegenwärtige Position innerhalb des BASIC-Programms im Hardware Stack (Stapelspeicher) abgelegt. Anschließend wird jeder Parameter hinter USR, (beginnend mit dem letzten) in eine hexadezimale Ganzzahl zwischen 0 und 65535 umgewandelt. Diese zwei-Byte Werte werden nun ebenfalls im Hardware Stack abgelegt. Dabei hat immer das niedrigwertige Byte Priorität vor dem höherwertigen. Der erste Parameter, der die Startadresse für das Maschinenprogramm spezifiziert, wird dagegen nicht im Hardware Stack eingetragen. Nachdem der letzte Wert auf dem Stack abgelegt wurde, fügt BASIC die Anzahl der zwei Byte-Parameter hinzu. Auch hierbei bleibt die Startadresse für das Maschinenprogramm unberücksichtigt. In Abbildung 11-4 wird demonstriert, wie die USR-Funktion mit dem Hardware Stack arbeitet. Die USR-Funktion spricht immer den 6502 Hardware Stack an, selbst wenn nur die Startadresse des Maschinenprogramms vorhanden ist. In diesem Fall wird nur die Anzahl der ein-Byte Parameter, welche Null lautet, auf dem Hardware Stack abgelegt. Bevor der Computer zum BASIC Programm zurückkehren kann, müssen diese Eintragungen aus dem

Hardware Stack wieder gelöscht sein. Der Befehl PLA aus der Assembler Sprache, kann Werte zwischen 0 und 65535 zurück zu einem BASIC-Programm übertragen. Dabei wird das niedrige Byte des Wertes in der Speicheradresse 212 abgelegt und das höhere Byte in Adresse 213. BASIC verwandelt die dort abgelegte hexadezimale Zahl in einen numerischen Wert. Das ergibt dann wiederum den Wert der USR-Funktion.
Um zurück zum Basic-Programm zu gelangen, muß das Maschinenprogramm mit RTS beendet werden. Dieser Assembler Befehl ist identisch mit dem Befehl 60 (hexadezimal), oder 96 (dezimal) in der Maschinensprache.

5

DER PROGRAMMRECORDER

Mit Hilfe des ATARI 410-Programmrecorders können Sie BASIC-Programme oder Daten auf Kassetten speichern. Zu einem späteren Zeitpunkt können diese Programme oder Daten wieder in den Computer eingelesen werden.

Die Programmspeicherung

In ATARI-BASIC gibt es drei Befehle, mit deren Hilfe Programme abgespeichert werden können: CSAVE, LIST und SAVE. Zu jedem dieser Befehle gibt es einen entsprechenden, mit dessen Hilfe die Programme wieder geladen werden können: CLOAD, ENTER und LOAD.

Abspeichern eines Programmes

Mit CSAVE können Programme nur auf Kassette abgelegt werden. Hingegen kann mit LIST oder SAVE ein Programm an irgendein Gerät ausgegeben werden, zum Beispiel an einen Drucker, an die Diskettenstation oder eben auch an den Programmrecorder. Mit jeder der folgenden Anweisungen kann ein Programm mit Hilfe des Programmrecorders auf Kassette gespeichert werden:

```
CSAVE
LIST "C:"
SAVE "C:"
```

Beachten Sie, daß bei LIST und SAVE angegeben werden muß, auf welchem Gerät die Ausgabe erfolgen soll. Das "C:" kennzeichnet den Programmrecorder. Mit CSAVE und SAVE "C:" kann ein Programm nur als Ganzes gespeichert werden. Mit LIST "C:" kann das ganze oder auch nur ein Teil des momentan im Speicher befindlichen Programmes abgelegt werden. Wie bei LIST üblich, geben Sie dann die erste und die letzte Zeilennummer an, die noch gespeichert werden soll. Wenn Sie zum Beispiel nur die Programmzeilen zwischen 100 und 1000 abspeichern wollen, geben Sie die folgende Anweisung:

```
LIST "C:",100,1000
```

Aber ganz gleich, ob Sie jetzt ein Programm mit CSAVE, LIST "C:" oder SAVE "C:" abspeichern – der Ablauf ist immer der gleiche: zunächst wird zweimal ein Kontrollton ertönen. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder zu legen und sie mit Hilfe der REWIND- (schneller Rücklauf) und ADVANCE-Taste (schneller Vorlauf) auf der Stelle zu positionieren, wo das Programm abgespeichert werden soll. Meistens wird das der Bandanfang sein. Dann drücken Sie die RECORD- und PLAY-Taste. Der ATARI-Computer kann nicht feststellen, ob Sie das alles getan haben. Sie müssen ihm das jetzt mitteilen, indem Sie die RETURN-Taste drücken. Daraufhin wird der Motor des Programmrecorders anlaufen. Wenn der Lautstärkerregler Ihres Fernsehgerätes aufgedreht ist, kön-

nen sie mithören, während das Programm abgespeichert wird. Zunächst wird ein klarer hoher Ton zu hören sein. Daraufhin folgen dann eine oder mehrere schnelle Tonfolgen. Mit jeder dieser Tonfolgen wird ein Teil des Programmes gespeichert. Je länger das Programm ist, in desto mehr Teile wird es aufgespalten und desto mehr Tonfolgen werden Sie hören. Wenn der Speichervorgang beendet ist, bleibt der Motor des Programmrecorders stehen. Sie können dann die STOP-Taste drücken.

Laden eines Programmes

Mit CLOAD können Programme geladen werden, die mit CSAVE auf Kassette abgelegt wurden. Mit ENTER oder LOAD können Programme von irgendeinem Gerät eingelesen werden. Mit ENTER "C:" können nur Programme geladen werden, die mit LIST "C:" gespeichert wurden und mit LOAD "C:" können nur Programme geladen werden, die mit SAVE "C:" gespeichert wurden.
Bei Ausführung des CLOAD- oder LOAD-Befehls wird das momentan im Speicher befindliche Programm (falls vorhanden) gelöscht und erst dann das neue Programm geladen. Bei Ausführung des ENTER-Befehls wird das neue Programm dem momentan im Speicher befindlichen überlagert. Falls das neue Programm Zeilen mit Zeilennummern enthält, die bereits bei dem im Speicher befindlichen Programm vorkommen, dann wird die alte Zeile durch die neue ersetzt. Wenn Sie diese Überlagerung vermeiden wollen, geben Sie einfach NEW ein, bevor Sie ein Programm mit ENTER laden.
Aber ganz gleich, ob Sie jetzt ein Programm mit CLOAD, ENTER "C:" oder LOAD "C:" laden – der Ablauf ist immer der gleiche: zunächst wird ein Kontrollton ertönen. Das ist für Sie das Zeichen, die Kassette mit dem Programm, das geladen werden soll, in den Programmrecorder einzulegen und sie mit Hilfe der REWIND- (schneller Rücklauf) und ADVANCE-Taste (schneller Vorlauf) auf der Stelle zu positionieren, wo sich das gewünschte Programm befindet. Meistens wird das der Bandanfang sein. Dann drücken Sie die PLAY-Taste. Der ATARI-Computer kann nicht feststellen, ob Sie das alles getan haben. Sie müssen ihm das jetzt mitteilen, indem Sie die RETURN-Taste drücken.
Daraufhin wird der Motor des Programmrecorders anlaufen. Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, können Sie mithören, während das Programm geladen wird. Zunächst werden Sie etwa 20 Sekunden lang nichts hören. Darauf folgen dann eine oder mehrere schnelle Tonfolgen. Mit jeder dieser Tonfolgen wird ein Teil des Programmes geladen. Je länger das Programm ist, aus desto mehr Teilen besteht es und desto mehr Tonfolgen werden Sie hören. Wenn der Ladevorgang beendet ist, bleibt der Motor des Programmrecorders stehen. Sie können dann die STOP-Taste drücken.

Das Zählwerk des Programmrecorders

Meistens werden Sie Ihre Programme am Anfang einer Kassette abspeichern. So ist es immer wieder leicht zu finden: Sie brauchen die Kassette nur zurückzuspulen. Nun kann ein Programm aber auch irgendwo anders auf der Kassette gespeichert werden. Aber wenn Sie es dann nicht wiederfinden, können Sie es auch nicht laden.
Das kann Ihnen allerdings nicht passieren, wenn Sie das Zählwerk des Programmrecorders benutzen. Sie müssen allerdings daran denken, daß Sie das Zählwerk zurücksetzen, wenn Sie die Kassette zurückspulen. Das Zählwerk muß 0 anzeigen, wenn die Kassette zurückgespult ist. Setzen Sie das Zählwerk niemals zu einem anderen Zeitpunkt zurück. Wenn Sie eine neue Kassette in den Programmrecorder einlegen, gehen Sie nicht davon aus, daß sie zurückgespult ist. Um sicher zu gehen, drücken Sie immer erst die REWIND-Taste und setzen Sie erst dann das Zählwerk zurück. Wenn Sie das alles beachten, werden Sie niemals das Problem haben, daß ein Programm nicht geladen werden kann, obwohl der Zählerstand doch derselbe ist, den Sie sich nach dem Abspeichern notiert hatten.

Die Geschwindigkeit des Zählwerkes variiert von Recorder zu Recorder. Zählerstände können also nicht ohne weiteres von einem auf den anderen Recorder übertragen werden.

Laden und Starten eines Programmes in einem Schritt

Mit Hilfe eines abgewandelten RUN-Befehls können Sie in einem Schritt ein Programm von der Kassette laden und starten. Dieser Befehl sieht folgendermaßen aus:

```
RUN "C:"
```

Im Prinzip handelt es sich hierbei um eine Kombination von LOAD "C:" und RUN. Deshalb funktioniert dieser Befehl auch nur bei Programmen, die mit SAVE "C:" abgespeichert wurden.

Verkettung von Programmen

Der RUN-Befehl kann sowohl direkt eingegeben als auch in Programmen verwendet werden. Wenn ein Programm eine Zeile mit der Anweisung RUN "C:" enthält und im Verlauf des Programms diese Zeile ausgeführt wird, dann wird ein neues Programm geladen und gestartet. Diesen Vorgang nennt man "CHAINING".
Ein kleines Beispiel soll das veranschaulichen. Legen Sie eine Kassette in den Programmrecorder, spulen Sie sie zurück und geben Sie folgendes ein:

```
NEW

READY
10 PRINT "ERSTES PROGRAMM"
20 PRINT "DRUECKEN SIE DIE PLAY-TASTE
AM PRO-"
30 PRINT "GRAMMRECORDER UND DANN RETUR
N!"
40 RUN "C:"
SAVE "C:"

READY
```

Damit wäre das erste Programm auf Kassette gepeichert. Es enthält in Zeile 20 und 30 eine Information für den Benutzer des Programmes, damit er weiß, was zu tun ist, wenn durch den RUN "C:"-Befehl in Zeile 40 ein Kontrollton erzeugt wird.
Das zweite Programm unterscheidet sich nur in einer Zeile vom ersten. Da sich das erste Programm noch im Speicher befindet, geben Sie einfach diese Zeile neu ein und speichern dann das Ergebnis auf Kassette:

```
10 PRINT "ZWEITES PROGRAMM"
SAVE "C:"

READY
```

Damit wäre auch das zweite Programm auf Kassette gespeichert. Für das dritte und letzte Programm sind wiederum nur einige Änderungen am noch im Speicher befindlichen zwei-

ten Programm vorzunehmen. Speichern Sie dann auch dieses Programm auf Kassette:

```
10 PRINT "DRITTES PROGRAMM"
20 END
30
40
SAVE "C:"

READY
```

Auf der Kassette sind nun drei Programme hintereinander gespeichert. Das erste Programm lädt und startet das zweite und das zweite lädt und startet das dritte. Spulen Sie die Kassette zurück und probieren Sie es aus:

```
RUN "C:"
ERSTES PROGRAMM
DRUECKEN SIE DIE PLAY-TASTE AM PRO-
GRAMMRECORDER UND DANN RETURN!
ZWEITES PROGRAMM
DRUECKEN SIE DIE PLAY-TASTE AM PRO-
GRAMMRECORDER UND DANN RETURN!
DRITTES PROGRAMM

READY
```

Auf diese Weise verkettete Programme sehen für den Anwender wie ein einziges langes Programm aus. Der Anwender muß allerdings jedesmal RETURN drücken, wenn ein neues Programm geladen werden soll. Im Ablauf des Programmes mag das manchmal als etwas störend empfunden werden, aber auf jeden Fall wäre es sehr viel störender, wenn statt dieses Tastendrucks jedesmal RUN "C:" eingegeben werden müßte.
Bei der Verkettung von Programmen mit Hilfe des RUN-Befehls gibt es allerdings ein Problem: Bevor das nächste Programm geladen wird, werden alle Variablen gelöscht. Aus diesem Grund kann ein Programm in einer solchen Kette niemals Variablen oder Ergebnisse von Berechnungen eines vorhergehenden Programmes verwenden.

Zusammenstellung von Unterprogrammen

Die in Kapitel 4 vorgestellten Unterprogramme (Bilder 4-16 bis 4-21, 4-31 und 4-36 bis 4-38) sind zweifellos sehr nützlich, aber andererseits ist es sehr unbequem, sie jedesmal neu einzutippen, wenn man sie braucht. Mit Hilfe des CSAVE- und des CLOAD-Befehls können Sie das jedoch umgehen. Geben Sie alle Unterprogramme auf einmal ein und speichern Sie diese Zusammenstellung mit CSAVE ab. Wenn Sie nun ein neues Programm schreiben, laden Sie mit CLOAD zunächst diese Zusammenstellung und löschen die Zeilen, die Sie nicht benötigen. Übrig bleiben dann die Unterprogramme, die Sie für die Erstellung Ihres Programmes brauchen. Diese Methode funktioniert, solange zwei Unterprogramme nicht dieselben Zeilennummern haben oder alle Unterprogramme zusammen nicht mehr genug Platz im Speicher haben.
Wenn Sie den LIST "C:"- und den ENTER "C:"-Befehl benutzen, ist es noch einfacher, bestehende Unterprogramme in ein neues Programm einzubauen. Zunächst müssen Sie jedes einzelne Unterprogramm mit Hilfe von LIST "C:" abspeichern. Am besten speichern Sie nur ein oder zwei Unterprogramme auf jeder Kassettenseite. Wenn Sie nun ein neues Programm schreiben, können Sie das gewünschte Unterprogramm jederzeit mit Hilfe von ENTER "C:" hinzuladen.

Die Aufzeichnungsarten

Offensichtlich scheint es ganz gleich zu sein, ob Sie ein Programm mit CSAVE, LIST "C:" oder SAVE "C:" abspeichern: Aber in Wirklichkeit ist die Aufzeichnungsart jedesmal anders.
Wenn Sie den LIST-Befehl verwenden, wird der ATASCII- Kode jedes Zeichens des Programmlistings ausgegeben, wobei es völlig gleichgültig ist, welches Gerät angesprochen wird.
Wenn ein Programm mit Hilfe von CSAVE oder SAVE gespeichert wird, wird jeder Befehl durch ein einzelnes Zeichen repräsentiert. Beim PRINT-Befehl wird an Stelle von fünf ATASCII-Zeichen dann nur noch ein Zeichen ausgegeben. Dieses Zeichen stellt dann praktisch eine Abkürzung für PRINT dar. Mit Hilfe von CLOAD und LOAD können Programme, die in dieser Aufzeichnungsart gespeichert wurden, wieder geladen werden. Da der Computer die Kodierung und Dekodierung übernimmt, dürfte es kaum von Interesse sein, welches Zeichen für welchen Befehl steht.
Bei der Speicherung werden Programme in Teilstücke zerlegt. Die Pause zwischen zwei Teilstücken ist bei CSAVE und SAVE "C:" unterschiedlich lang: Bei CSAVE ist sie etwas größer als bei SAVE "C:". Aus diesem Grund geht das Speichern und Laden von Programmen bei Benutzung von CSAVE UND CLOAD schneller als bei Benutzung von SAVE "C:" und LOAD "C:".

Die Variablentabelle

ATARI-BASIC trägt alle Variablen- und Feldnamen in eine Tabelle ein. Dabei spielt es keine Rolle, ob diese Variablen oder Felder in einem Programm oder im Eingabemodus verwendet wurden. Wenn ein Programm mit Hilfe von CSAVE oder SAVE gespeichert wird, dann wird diese Variablentabelle mitgespeichert. Zusammen mit dem Programm wird diese Variablentabelle bei Ausführung von CLOAD oder LOAD wieder geladen, wobei sie die im Speicher vorhandene Variablentabelle ersetzt. Beim Speichern von Programmen mit Hilfe des LIST-Befehls wird keine Variablentabelle gespeichert. Also kann bei Ausführung des ENTER-Befehls auch keine Variablentabelle geladen werden. Die vorhandene Variablentabelle bleibt bestehen und bei Ausführung des Programmes werden alle neuen Variablen und Felder zusätzlich eingetragen.
Wenn Sie längere Zeit an einem Programm gearbeitet haben, werden in der Variablentabelle einige Variablen eingetragen sein, die Sie gar nicht mehr benötigen. Um diese Variablen zu beseitigen, gehen Sie folgendermaßen vor: speichern Sie das Programm mit Hilfe des LIST-Befehls. Löschen Sie die Variablentabelle, indem Sie NEW eingeben. Da das Programm dabei auch gelöscht wurde, laden Sie es jetzt wieder mit Hilfe des ENTER-Befehls.

Die Datenspeicherung

Sehr oft muß ein Computer große Datenmengen verarbeiten. Unter Umständen sind diese Datenmengen so groß, daß nicht alle Daten auf einmal im Speicher festgehalten werden können. In ATARI-BASIC können Sie mit Hilfe des PRINT #- und des PUT-Befehls Daten auf Kassette speichern. Mit Hilfe des INPUT #- und des GET-Befehls können diese Daten wieder geladen werden.

Datenfiles, Records und Fields

Der Computer speichert Daten auf Kassette in sogenannten FILES. Wenn die Kassette einem Aktenschrank entspricht, dann sind die Files die Schubladen dieses Aktenschran-

kes. Genauso wie ein Aktenschrank eine oder mehrere Schubladen haben kann, können auf einer Kassette ein oder mehrere Files gespeichert sein. Und genauso wie die Schublade eines Aktenschrankes kann auch ein Datenfile auf Kassette voll oder leer sein.
Die Files werden in RECORDS und FIELDS unterteilt. Sie können mit den Schnellheftern in einer Schublade des Aktenschrankes und deren Inhalt verglichen werden. Die Anzahl von Records in einem Datenfile kann beliebig groß sein. Eine Begrenzung ist hier nur durch die Bandlänge gegeben. Die Anzahl der Fields in einem Record ist ebenfalls beliebig. Im allgemeinen wird die Anzahl der Fields in den Records eines Files jedoch gleich groß sein. Wenn ein Field nicht benutzt wird, ist sein Wert Null beziehungsweise es ist leer. Um das Ende eines Files zu markieren, schreibt der Computer einen speziellen Record, den sogenannten "End-of-File"-Record (EOF).
Einen Nachteil haben auf Kassette angelegte Datenfiles jedoch: auf sie kann nur sequentiell zugegriffen werden. Das heißt, Sie beginnen stets am Anfang eines Files und lesen Record für Record bis zum Ende. Hinzufügen oder Löschen von Records ist nicht möglich.

Der Kassettenbuffer

Obwohl die Datenübertragung vom und zum Programmrecorder im Prinzip zeichenweise erfolgt, überträgt der ATARI-Computer Blöcke mit je 128 Zeichen. Ein Teil des Speichers ist dafür reserviert, diese 128 Zeichen zwischenzuspeichern. Diesen Speicherbereich nennt man den Kassettenbuffer.

Das Format von Kassettenfiles

Jedes auf Kassette abgespeicherte Datenfile besteht aus drei Teilen: dem sogenannten "Leader", den Datenblöcken und dem End-of-File-Record (Bild 5-1). Der 20 Sekunden dauernde Leader dient dazu, Programmrecorder und Computer zu synchronisieren und für die Datenübertragung vorzubereiten. Alle Datenblöcke, mit Ausnahme des letzten, enthalten 128 Zeichen (Bytes) an Information. Der letzte Datenblock enthält die letzten Zeichen des Files. Im Normalfall sind das weniger als 128. Der allerletzte Block eines Files enthält den End-of-File-Record.
Abhängig von seiner Länge kann ein Record genau einen, mehrere oder weniger als einen Block beanspruchen. Bei der Programmierung braucht darauf keine Rücksicht genommen werden, weil der Computer diese Dinge automatisch berücksichtigt. Die einzige Ausnahme wird im nächsten Abschnitt behandelt.

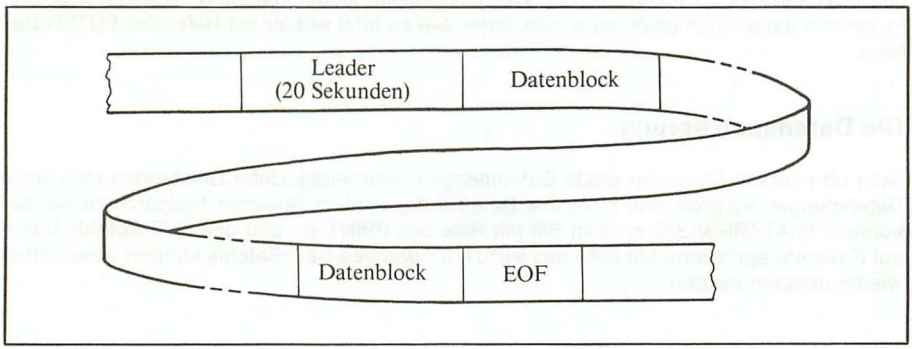


Bild 5-1: Aufbau eines Kassettenfiles

Öffnen eines Datenfiles

Wenn ein File geöffnet wird, kann auf die Information in diesem File zugegriffen werden und zwar so lange, bis das File wieder geschlossen wird. Die folgende Anweisung öffnet ein Datenfile auf der Kassette mit Hilfe des OPEN-Befehls:

```
OPEN #1,8,0,"C:"
```

Damit ist Kanal 1 für Ausgabe auf den Programmrecorder geöffnet. Genauso wie bei anderen Peripheriegeräten auch wird durch die zweite Zahl nach dem OPEN-Befehl festgelegt, ob der angesprochene Kanal als Ein- oder Ausgabekanal verwendet wird. Eine 4 steht für Eingabe und eine 8 für Ausgabe. Beim Programmrecorder ist es nicht möglich, ein File gleichzeitig für Ein- und Ausgabe zu öffnen. Die dritte Zahl nach dem OPEN-Befehl ist normalerweise immer eine 0.
Wenn Sie ein Kassettenfile öffnen, geschieht zunächst das gleiche wie beim Speichern oder Laden eines Programmes: wenn Daten gelesen werden sollen, ertönt ein Kontrollton und wenn Daten geschrieben werden sollen, ertönen zwei Kontrolltöne. Der Computer wartet dann, bis Sie die Kassette mit Hilfe der REWIND- und der ADVANCE-Taste an der richtigen Stelle positioniert und die RETURN-Taste gedrückt haben. Danach wird der Motor des Programmrecorders anlaufen.
Besonders anwenderfreundlich ist es, wenn eine Meldung auf dem Bildschirm erscheint, bevor ein Kassettenfile geöffnet wird. Der Anwender wird dann nicht plötzlich von einem Kontrollton überrascht. Ein Beispiel dafür wäre folgendes:

```
10 PRINT "Positionieren Sie die Kasset
te und"
20 PRINT "druecken Sie RETURN!"
30 OPEN #2,4,0,"C:"
```

Sobald der Motor des Programmrecorders angelaufen ist, beginnt der Computer mit dem Lesen oder Schreiben des Leaders. Wenn Sie einen Ausgabekanal geöffnet haben, wird der Leader geschrieben. Wenn es sich um einen Eingabekanal handelt, wird der Leader gelesen. Während dieses Vorganges können keine anderen Befehle ausgeführt werden.
Sobald der Leader gelesen beziehungsweise geschrieben wurde, müssen vom Programm aus sofort die ersten Daten gelesen beziehungsweise geschrieben werden. Wenn es sich um einen Ausgabekanal handelt, müssen 128 Zeichen an den Programmrecorder ausgegeben werden. Falls zu diesem Zeitpunkt gerade keine Daten anfallen, kann ein Block geschrieben werden, der nur aus Nullen beziehungsweise Leerzeichen besteht. Wenn das nicht getan wird, kann beim späteren Lesen des Files ein Fehler auftreten. Falls es sich um einen Eingabekanal handelt, müssen die ersten Daten des Files gelesen werden. Auch hier kann beim späteren Lesen ein Fehler auftreten, wenn das nicht getan wird. Beispiele werden noch behandelt.

Schließen eines Datenfiles

Wenn ein Datenfile nicht mehr angesprochen wird, sollte es stets geschlossen werden, weil der entsprechende Ein/Ausgabekanal dann für andere Zwecke frei wird. Das Schließen eines Datenfiles erfolgt mit Hilfe des CLOSE-Befehls. Ein Beispiel dafür ist die folgende Anweisung:

```
CLOSE #1
```

Das Schließen eines Datenfiles ist besonders wichtig, wenn Daten ausgegeben wurden. Anderenfalls könnten Daten verlorengehen. Nehmen Sie an, daß der Kassettenbuffer teil-

weise mit Daten gefüllt ist. Wenn das Datenfile geschlossen wird, wird dieser teilweise gefüllte Buffer als letzter Datenblock auf die Kassette geschrieben. Wenn das File nicht geschlossen wird, wird der Buffer nicht mehr ausgegeben und die Daten somit nicht gespeichert.
Bei Ausführung des END- oder des RUN-Befehls werden automatisch alle geöffneten Kanäle geschlossen.

Das Beschreiben von Datenfiles

Mit Hilfe des PRINT #- oder des PUT-Befehls können Daten über einen vorher geöffneten Kanal ausgegeben werden. Dabei spielt es keine Rolle, welches Gerät durch diesen Kanal angesprochen wird. Wenn dem Kanal mit Hilfe des OPEN-Befehls der Programmrecorder (C:) zugeordnet wurde, dann werden die Daten dorthin übertragen. Das folgende Programm soll diesen Sachverhalt veranschaulichen:

```
10 PRINT "Positionieren Sie die Kasset
te und"
20 PRINT "druecken Sie RETURN!"
30 OPEN #3,8,0,"C:"
40 PRINT #3;"Dieser Satz wird an den P
rogrammrecorder uebertragen"
50 CLOSE #3
```

Bei der Ausgabe von Zahlen oder Strings mit Hilfe des PRINT #- Befehls wird der ATASCII-Kode benutzt. Benutzen Sie bei Verwendung des PRINT #- Befehls zur Trennung stets Semikolons an Stelle von Kommas sind zwar zuverlässig, aber bei ihrer Verwendung werden zusätzliche Leerzeichen aufgezeichnet.
Jeder Datenwert, der an den Programmrecorder übertragen wird, muß von einem EOL- (End-of-Line = Zeilenende) Zeichen gefolgt sein. Wenn die Daten wieder gelesen werden, kann so festgestellt werden, wo ein Datenwert endet und der nächste beginnt. Um sicherzustellen, daß das EOL-Zeichen nach jedem Datenwert ausgegeben wird, gibt es zwei Möglichkeiten: Erstens kann jeder Datenwert mit einem eigenen PRINT #-Befehl ausgegeben werden. Am Ende eines solchen Befehls darf dann aber niemals ein Semikolon oder ein Komma stehen. Zweitens kann das EOL-Zeichen auch mit Hilfe von CHR$ erzeugt werden:

```
10 PRINT "Positionieren Sie die Kasset
te und"
20 PRINT "druecken Sie RETURN!"
30 OPEN #3,8,0,"C:"
40 PRINT #3;A;CHR$(155);B;CHR$(155);C
50 CLOSE #3
```

Mit Hilfe des PUT-Befehls kann eine Zahl zwischen 0 und 255 ausgegeben werden. Diese Zahl belegt dann genausoviel Platz wie ein einziges ATASCII-Zeichen. Das folgende Programm gibt sofort nachdem ein Ausgabekanal für den Programmrecorder geöffnet wurde, einen Block aus, der nur aus Nullen besteht:

```
10 PRINT "Positionieren Sie die Kasset
te und"
20 PRINT "druecken Sie RETURN!"
30 OPEN #3,8,0,"C:"
35 REM Leerer Block
40 FOR J=1 TO 128
```

```
50 PUT #3,0
60 NEXT J
70 PRINT #3;"Jetzt kommen die Daten"
80 CLOSE #3
```

Das Lesen von Datenfiles

Mit Hilfe des INPUT #-Befehls können Daten gelesen werden, die mit Hilfe des PRINT #-Befehls geschrieben wurden. Der Kanal, der durch diesen Befehl angesprochen wird, muß ein Eingabekanal sein. Um die Funktion dieses Befehls kennenzulernen, erzeugen Sie zunächst mit Hilfe des folgenden Programmes ein Datenfile:

```
10 DIM A$(120)
20 PRINT "Geben Sie die zu speichernde
n Daten"
30 PRINT "ein."
40 INPUT A$
50 PRINT
60 PRINT "Positionieren Sie die Kasset
te und"
70 PRINT "druecken Sie RETURN!"
80 OPEN #3,8,0,"C:"
90 PRINT #3;A$
100 CLOSE #3
```

Das nächste Programm liest die gespeicherten Daten mit Hilfe des INPUT #-Befehls. Vergessen Sie nicht, die Kassette zurückzuspulen.

```
10 DIM A$(120)
20 PRINT "Positionieren Sie die Kasset
te und"
30 PRINT "druecken Sie RETURN!"
40 OPEN #3,4,0,"C:"
50 INPUT #3;A$
60 PRINT A$
70 CLOSE #3
```

Die mit Hilfe des INPUT #-Befehls gelesenen Daten des Kassettenfiles werden als ATASCII-Kode interpretiert. Jedesmal wenn ein EOL-Zeichen (ATASCII-Kode 155) auftritt, werden die seit dem letzten EOL gelesenen Zeichen der angegebenen Variablen zugewiesen. Falls es sich dabei um eine numerische Variable handelt, werden die eingelesenen Zeichen in eine Zahl umgewandelt. Wenn das nicht möglich ist, tritt ERROR 8 auf.
Mit jedem GET-Befehl wird eine Zahl zwischen 0 und 255 gelesen. Diese Zahl wird dann auf irgendeine Art vom Programm interpretiert. Mit Hilfe von CHR$ können diese Daten beispielsweise als ATASCII-Zeichen interpretiert werden. Das folgende Programm liest dasselbe File wie das vorhergehende mit Hilfe des GET-Befehls:

```
10 PRINT "Positionieren Sie die Kasset
te und"
20 PRINT "druecken Sie RETURN!"
30 OPEN #3,4,0,"C:"
40 GET #3,A
45 REM Zeichen ausgeben bis EOL
50 IF A<>155 THEN ? CHR$(A);:GOTO 40
60 ? :REM Zeile beenden
70 CLOSE #3
```

Beachten Sie, daß hier das Programm selbst feststellen muß, ob ein EOL-Zeichen gelesen wurde (Zeile 50).

Ein Programmbeispiel

Eine Adressenverwaltung wird in der Praxis sehr oft benötigt. Dazu werden zwei Programme geschrieben. Mit Hilfe des ersten können Namen und Adressen eingegeben und in einem Kassettenfile gespeichert werden. Das zweite Programm liest diese Namen und Adressen und bringt sie wieder auf den Bildschirm. In Kapitel 6 wird ein weiteres Programm erstellt. Dieses Programm druckt mit Hilfe der von der Kassette gelesenen Namen und Adressen Adressenaufkleber.
Ein Record besteht aus vier Fields: dem Namen, der Straßenangabe, der Postleitzahl und dem Wohnort. In unserem Programm werden für diese Fields immer dieselben Variablen verwendet: NAME$, ST$, PLZ$ und WOHN$.
Das erste Programm dient der Erzeugung des Files. Zunächst müssen in diesem Programm die obigen Variablen dimensioniert werden. Dann wird ein Ausgabekanal für den Programmrecorder bereitgestellt. Da der Anwender die Daten eingeben muß, bevor sie gespeichert werden können, gibt das Programm nach dem Öffnen des Ausgabekanals zunächst einen leeren Block aus. In Programmform sieht das dann folgendermaßen aus:

```
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20),EOL$(1)
60 EOL$=CHR$(155):REM EOL-Zeichen
65 REM File oeffnen und leeren Block s
chreiben
70 GOSUB 3000
1900 END
2999 REM File fuer Programmrecorder oe
ffnen
3000 PRINT CHR$(125);"Positionieren Si
e die Kassette,"
3004 PRINT "druecken Sie RETURN und wa
rten Sie"
3006 PRINT "bitte einen Moment."
3010 OPEN #1,8,0,"C:"
3015 REM Leerblock schreiben
3020 FOR J1=1 TO 128:PUT #1,0:NEXT J1
3030 RETURN
```

Dann muß das Programm die Eingaben für einen Record entgegennehmen:

```
1000 PRINT CHR$(125);"Geben Sie Name u
nd Adresse ein"
1010 PRINT:PRINT
1020 PRINT "  Name:";
1030 INPUT NAME$
1040 PRINT "Strasse:";
1050 INPUT ST$
1060 PRINT "  PLZ:";
1070 INPUT PLZ$
1080 PRINT "Wohnort:";
1090 INPUT WOHN$
```

Es sollte möglich sein, die Eingabe zu wiederholen, falls ein Fehler gemacht wurde. Diese Aufgabe übernehmen die folgenden Programmzeilen:

```
1130 POSITION 2,1
1140 PRINT "Eingabe wiederholen?"
1150 INPUT R$;R$=R$(1,1)
1160 IF R$="N" OR R$="n" THEN 1300
1170 IF R$="J" OR R$="j" THEN 1000
1180 GOTO 1130
```

Wenn der Anwender angibt, daß die Daten fehlerfrei sind, können sie an den Programmrecorder übergeben werden. Diese Ausgabe erfolgt über den Kassettenbuffer. Falls im Kassetenbuffer noch Platz genug für alle Daten vorhanden ist, erfolgt die Ausgabe sehr schnell. Sobald der Kassettenbuffer voll geworden ist, muß er auf der Kassette gespeichert werden und das erfordert einige Sekunden Zeit. Um den Anwender nicht zu verunsichern, sollte er darauf hingewiesen werden, daß er einige Sekunden zu warten hat. Geben Sie die folgenden Zeilen ein:

```
1300 POSITION 8,21
1310 PRINT "* * Warten Sie bitte einen
 Moment * *";
1315 REM Ausgabe eines Records
1320 PRINT #1;NAME$;EOL$;ST$;EOL$;PLZ$
;EOL$;WOHN$
```

Wenn die Daten ausgegeben worden sind, kann die Meldung wieder gelöscht werden. Dann muß der Anwender darüber befragt werden, ob weitere Daten eingegeben werden sollen. Diese Aufgabe übernimmt der folgende Programmteil:

```
1325 REM Meldung loeschen
1330 PRINT CHR$(156);REM Zeile loesche
n
1340 POSITION 2,1
1350 PRINT "Wollen Sie noch mehr Adres
sen ein-"
1355 PRINT "geben?"
1360 INPUT R$;R$=R$(1,1)
1370 IF R$="J" or R$="j" THEN 1000
1380 IF R$<>"N" AND R$<>"n" THEN 1340
```

Falls keine weiteren Daten eingegeben werden sollen, schreibt das Programm noch einen zusätzlichen Record, den sogenannten TRAILER und schließt dann das File. Der Trailer kennzeichnet das Ende des Files. Im Trailer sind ganz bestimmte Worte gespeichert – Werte, die durch eine Eingabe des Anwenders niemals zustande kommen können. Ein Programm, daß dieses File liest, kann die gelesenen Daten auf diese Werte hin untersuchen und bei ihrem Auftreten den Lesevorgang stoppen. Die folgenden Zeilen beenden das Programm:

```
1385 REM Ausgabe des Trailers
1390 FOR J=1 TO 4;REM 4 Fields
1400 PRINT #1;CHR$(253);EOL$;
1410 NEXT J
1420 CLOSE #1
1900 END
```

```
*50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
*(20),R$(20),EOL$(1)
*60 EOL$=CHR$(155):REM EOL-Zeichen
*65 REM File oeffnen und leeren Block s
*chreiben
*70 GOSUB 3000
1000 PRINT CHR$(125);"Geben Sie Name u
nd Adresse ein"
1010 PRINT:PRINT
1020 PRINT "  Name:";
1030 INPUT NAME$
1040 PRINT "Strasse:";
1050 INPUT ST$
1060 PRINT "  PLZ:";
1070 INPUT PLZ$
1080 PRINT "Wohnort:";
1090 INPUT WOHN$
1130 POSITION 2,1
1140 PRINT "Eingabe wiederholen?"
1150 INPUT R$:R$=R$(1,1)
*1160 IF R$="N" OR R$="n" THEN 1300
1170 IF R$="J" OR R$="j" THEN 1000
1180 GOTO 1130
*1300 POSITION 8,21
*1310 PRINT "* * Warten Sie bitte einen
* Moment * *";
*1315 REM Ausgabe eines Records
*1320 PRINT #1;NAME$;EOL$;ST$;EOL$;PLZ$
*;EOL$;WOHN$
*1325 REM Meldung loeschen
*1330 PRINT CHR$(156):REM Zeile loesche
*n
*1340 POSITION 2,1
*1350 PRINT "Wollen Sie noch mehr Adres
*sen ein-"
*1355 PRINT "geben?"
*1360 INPUT R$:R$=R$(1,1)
*1370 IF R$="J" or R$="j" THEN 1000
*1380 IF R$<>"N" AND R$<>"n" THEN 1340
*1385 REM Ausgabe des Trailers
*1390 FOR J=1 TO 4:REM 4 Fields
*1400 PRINT #1;CHR$(253);EOL$;
*1410 NEXT J
*1420 CLOSE #1
1900 END
*2999 REM File fuer Programmrecorder oe
*ffnen
*3000 PRINT CHR$(125);"Positionieren Si
*e die Kassette,"
```

Anmerkung: Dieses Programm veranschaulicht die Erzeugung von Kassettenfiles. Zeilen, die vorne mit einem Stern versehen sind, können bei dem in Bild 4-25 aufgeführten Programm eingefügt werden. Dann werden auch bei diesem Programm die eingegebenen Daten auf Kassette gespeichert.

Bild 5-2: Adressenverwaltungsprogramm/Eingabe

```
*3004 PRINT "druecken Sie RETURN und wa
*rten Sie"
*3006 PRINT "bitte einen Moment."
*3010 OPEN #1,8,0,"C:"
*3015 REM Leerblock schreiben
*3020 FOR J1=1 TO 128:PUT #1,0:NEXT J1
*3030 RETURN
```

Bild 5-2: Adressenverwaltungsprogramm/Eingabe (Fortsetzung)

In Bild 5-2 ist das Programm noch einmal vollständig aufgelistet. Sicher haben Sie festgestellt, daß die Dateneingabe nicht besonders komfortabel ist. Bei einem der in Kapitel 4 erstellten Programme (Bild 4-25) werden dieselben Daten (Name und Adresse) eingegeben, aber der Teil des Programmes, der die Eingaben verarbeitet, ist sehr viel umfangreicher als bei dem vorliegenden Programm. Die Zeilen 1000 bis 1180 in Bild 5-2 erfüllen im Prinzip die gleiche Funktion wie die Zeilen 1000 bis 1230 in Bild 4-25. Um das Adressenverwaltungsprogramm zu optimieren, können Sie die Teile des Programmes in Bild 5-2, mit deren Hilfe die eingegebenen Daten auf Kassette gespeichert werden (Zeile 50-70, 1160, 1300-1420 und 3000-3030) mit dem in Bild 4-25 aufgeführten Programm kombinieren (das ganze Programm außer Zeile 50 und 1160).
Ein Programm, das die gespeicherten Daten wieder lesen soll, muß zunächst die für diesen Zweck benötigten Variablen dimensionieren. Dann muß ein Eingabekanal für den Programmrecorder geöffnet und der Leerblock gelesen werden:

```
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20)
65 REM File oeffnen und Leerblock lese
n
70 GOSUB 3100
1900 END
3099 REM File fuer Programmrecorder oe
ffnen
3100 PRINT CHR$(125);"Positionieren Si
e die Kassette,"
3104 PRINT "druecken Sie RETURN und wa
rten Sie"
3106 PRINT "bitte einen Moment."
3110 OPEN #1,4,0,"C:"
3115 REM Leerblock lesen
3120 FOR J1=1 TO 128:GET #1,R:NEXT J1
3130 RETURN
```

Nun können die Daten gelesen werden. Vorher werden allerdings noch einige Informationen für den Anwender zur Anzeige gebracht. Das Stoppen der Bildschirmausgabe durch Drücken von CTRL-1 ist eine Funktion, die im ATARI-Computer fest eingebaut ist. Durch erneutes Drücken von CTRL-1 wird die Bildschirmausgabe wieder fortgesetzt. Die folgenden Zeilen teilen das dem Anwender mit:

```
1000 PRINT CHR$(125):REM Bildschirm lo
eschen
1010 PRINT " ANZEIGE DER GESPEICHERTEN
 ADRESSEN"
```

```
1020 POSITION 2,20
1030 PRINT "Druecken Sie CTRL-1, um di
e Ausgabe zu"
1035 PRINT "stoppen. Um die Ausgabe fo
rtzusetzen,"
1040 PRINT "druecken Sie CTRL-1 ein zw
eites Mal."
1050 POSITION 2,2
1060 PRINT "Druecken Sie RETURN, wenn
mit der"
1065 PRINT "Ausgabe begonnen werden so
ll."
1070 INPUT R$
```

Die Adressen sollen auf dem Bildschirm ausgegeben werden. Aus diesem Grund öffnet das Programm als nächstes einen Ausgabekanal für den Bildschirm. Dann wird ein Record nach dem anderen gelesen und auf dem Bildschirm zur Anzeige gebracht. Nach jedem Lesevorgang wird geprüft, ob es sich bei den gelesenen Daten um den Trailer handelt. Ohne diese Prüfung würde früher oder später ein EOF-Zeichen gelesen, was einen Fehler (Error 136) zur Folge hätte. Mit den folgenden Zeilen wird das Programm beendet:

```
1075 REM Adressen anzeigen
1080 OPEN #4,8,0,"S:"
1085 REM Naechsten Record lesen
1090 INPUT #1,NAME$,ST$,PLZ$,WOHN$
1095 REM Ueberpruefen, ob Trailer gele
sen wurde
1100 IF NAME$=CHR$(253) THEN 1200
1110 PRINT #4;NAME$
1120 PRINT #4;ST$
1130 PRINT #4;PLZ$;" ";WOHN$
1150 PRINT:PRINT
1160 GOTO 1090
1199 REM Trailer wurde gelesen/Program
m beenden
1200 CLOSE #1
1210 CLOSE #4
```

```
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20)
65 REM File oeffnen und Leerblock lese
n
70 GOSUB 3100
1000 PRINT CHR$(125):REM Bildschirm lo
eschen
1010 PRINT " ANZEIGE DER GESPEICHERTEN
 ADRESSEN"
1020 POSITION 2,20
1030 PRINT "Druecken Sie CTRL-1, um di
e Ausgabe zu"
```

Bild 5-3: Adressenverwaltungsprogramm/Anzeige

```
1035 PRINT "stoppen. Um die Ausgabe fo
rtzusetzen,"
1040 PRINT "druecken Sie CTRL-1 ein zw
eites Mal."
1050 POSITION 2,2
1060 PRINT "Druecken Sie RETURN, wenn
mit der"
1065 PRINT "Ausgabe begonnen werden so
ll."
1070 INPUT R$
1075 REM Adressen anzeigen
1080 OPEN #4,8,0,"S:"
1085 REM Naechsten Record lesen
1090 INPUT #1,NAME$,ST$,PLZ$,WOHN$
1095 REM Ueberpruefen, ob Trailer gele
sen wurde
1100 IF NAME$=CHR$(253) THEN 1200
1110 PRINT #4;NAME$
1120 PRINT #4;ST$
1130 PRINT #4;PLZ$;" ";WOHN$
1150 PRINT:PRINT
1160 GOTO 1090
1199 REM Trailer wurde gelesen/Program
m beenden
1200 CLOSE #1
1210 CLOSE #4
1900 END
3099 REM File fuer Programmrecorder oe
ffnen
3100 PRINT CHR$(125);"Positionieren Si
e die Kassette,"
3104 PRINT "druecken Sie RETURN und wa
rten Sie"
3106 PRINT "bitte einen Moment."
3110 OPEN #1,4,0,"C:"
3115 REM Leerblock lesen
3120 FOR J1=1 TO 128:GET #1,R:NEXT J1
3130 RETURN
```

Anmerkung: Dieses Programm veranschaulicht das Lesen von Kassettenfiles. Es ist möglich, dieses Programm so zu modifizieren, daß die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker erfolgt (siehe Bild 6-4).

Bild 5-3: Adressenverwaltungsprogramm/Anzeige (Fortsetzung)

In Bild 5-3 ist das Programm noch einmal vollständig aufgelistet. In Kapitel 6 wird dieses Programm so modifiziert, daß die Ausgabe nicht mehr auf dem Bildschirm sondern auf dem Drucker erfolgt (Bild 6-4).

6

DIE ATARI-DRUCKER

Nach dem Einschalten des ATARI-Computers erfolgen alle Ausgaben auf dem Bildschirm. Es ist aber sehr leicht möglich, statt des Bildschirms einen Drucker anzusprechen.
In diesem Kapitel wird hauptsächlich die Frage behandelt werden, wie man die drei ATARI-Drucker ATARI 820 (zur Zeit in Deutschland nicht erhältlich) ATARI 822 und ATARI 825 (entspricht Centronics 737-2) vom Programm aus ansprechen kann. Wenn normaler Text oder Zahlen ausgedruckt werden sollen, gibt es bei den drei Druckern nur wenig Unterschiede. Der größte Unterschied besteht noch in der unterschiedlichen Zeilenlänge. Die ATARI-Drucker 820 und 822 können 40 Zeichen pro Zeile drucken. Der ATARI-Drucker 825 druckt normalerweise 80 Zeichen pro Zeile; maximal sind jedoch 132 Zeichen pro Zeile möglich. Er bietet außerdem noch einige Sonderfunktionen, die am Ende dieses Kapitels behandelt werden. Falls nichts anderes angegeben ist, werden bis dahin alle drei Drucker genauso angesprochen.
Bevor Sie weiterlesen, überprüfen Sie, ob Ihr Drucker eingeschaltet und richtig angeschlossen ist. Der ATARI-Drucker 825 muß über das ATARI-Interfacemodul 850 angeschlossen werden. In diesem Fall müssen selbstverständlich beide Geräte eingeschaltet sein. Näheres über den Anschluß der verschiedenen Geräte finden Sie in den jeweiligen Gerätebeschreibungen.

Ausdruck von Programmlistings

Mit Hilfe des LIST-Befehls kann das gerade im Speicher des ATARI-Computers befindliche Programm auf dem Bildschirm angegeben werden. Um dieses Listing stattdessen auf einem Drucker auszugeben, ist nur eine kleine Änderung erforderlich:

```
LIST "P:"
```

P: ist der Gerätename des Druckers. Durch die Angabe des Gerätenamens direkt hinter dem LIST-Befehl wird dem Computer mitgeteilt, wo die Ausgabe zu erfolgen hat. Es ist ebenfalls möglich, die erste und die letzte Zeile anzugeben, die ausgedruckt werden soll. Mit Hilfe der folgenden Anweisung werden alle Zeilen mit Zeilennummern zwischen 10 und 100 ausgedruckt:

```
LIST "P:",10,100
```

Die äußere Form des Listings ist immer gleich, unabhängig davon, auf welchem Gerät die Ausgabe erfolgt. Eine Ausnahme bildet allerdings die Zeilenlänge. Die maximale Zeilenlänge bestimmt das angesprochene Gerät und nicht der LIST-Befehl. Beim Bildschirm und den ATARI-Druckern 820 und 822 beträgt die maximale Zeilenlänge 40 Zeichen. Falls eine Programmzeile länger als 40 Zeichen ist, wird die nächste Drucker- beziehungsweise Bildschirmzeile begonnen. Beim ATARI-Drucker 825 beträgt die Zeilenlänge normalerweise 80 Zeichen. Dort wird die nächste Druckerzeile erst dann begonnen, wenn eine Programmzeile länger als 80 Zeichen ist.
Mit keinem Drucker können Grafikzeichen wie zum Beispiel oder ausgedruckt werden. Bei den ATARI-Druckern 820 und 822 wird an Stelle dieser Graphikzeichen ein Leerzeichen ausgegeben. Beim ATARI-Drucker 825 werden manche Graphikzeichen überhaupt nicht gedruckt, während einige andere Sonderfunktionen auslösen. Diese Sonderfunktionen werden am Ende dieses Kapitels beschrieben.

Ansprechen des Druckers über ein Programm

Das Ansprechen des Druckers über ein Programm funktinioniert fast genauso wie das Ansprechen des Bildschirms über ein Programm. Auf jeden Fall ist es bestimmt nicht schwieriger, obwohl es einige Unterschiede gibt. Zum Beispiel gibt es beim Drucker keinen Cursor. Mit Hilfe des POSITION-Befehls kann der Cursor auf jede beliebige Stelle des Bildschirms gesetzt werden, aber der Druckkopf eines Druckers kann damit nicht bewegt werden. Ein Drucker kann nur ganze Zeilen drucken und zwar nacheinander, eine nach der anderen. Auf dem Bildschirm kann ein "Format" erstellt werden, zum Beispiel in Form eines Fragebogens oder eines Formblattes (siehe Bild 4-24). Später können dann an den entsprechenden Stellen Einträge gemacht werden. Mit einem Drucker ist so etwas nicht möglich. Hier muß stattdessen eine Zeile komplett, das heißt in diesem Fall bestehend aus Format und entsprechendem Eintrag, ausgedruckt werden, bevor zur nächsten Zeile übergegangen werden kann.

Der LPRINT-Befehl

In ATARI-BASIC gibt es einen speziellen Befehl für die Benutzung des Druckers. Dieser Befehl heißt LPRINT. Mit ihm kann ein 40-Zeichen-Drucker auf genau dieselbe Weise angesprochen werden wie der Bildschirm mit PRINT. An Stelle des Bildschirms erfolgt bei Verwendung von LPRINT die Ausgabe auf dem Drucker. Hier ein Beispiel:

```
10 LPRINT "ANWENDUNG DES LPRINT-BEFEHL
S"
20 LPRINT
30 LPRINT "NR. EINS","NR. ZWEI","NR. D
REI"
40 LPRINT 2.2E+44,-100.76,
50 LPRINT 1234567890
```

Nicht ganz einwandfrei funktioniert der LPRINT-Befehl mit dem ATARI-Drucker 825. Wenn mit einem LPRINT-Befehl 40 Zeichen oder weniger ausgedruckt werden und dieser Befehl mit einem Semikolon oder einem Komma endet, dann wird bei Auftreten des nächsten LPRINT-Befehls ab der 41. Stelle derselben Druckzeile weitergedruckt. Wenn Sie das oben aufgeführte Programm laufen lassen, wird genau das bei Ausführung der Zeilen 40 und 50 passieren. Wenn jedoch zwischen 41 und 80 Zeichen ausgedruckt werden und der LPRINT-Befehl mit einem Semikolon oder einem Komma endet, dann wird mit dem näch-

sten LPRINT am Anfang der nachfolgenden Zeile weitergedruckt.
LPRINT benutzt Kanal 7 als Ausgabekanal zum Drucker. Beim Auftreten von LPRINT wird dieser Kanal automatisch geöffnet. Es ist also nicht notwendig, einen OPEN-Befehl zu geben. Wenn dieser Kanal jedoch bereits belegt ist, tritt ein Fehler auf und Kanal 7 wird geschlossen. Bei mehreren aufeinander folgenden LPRINT-Befehlen treten keine Probleme auf.

Verwendung von PRINT # und PUT zusammen mit einem Drucker

Sowohl mit dem PRINT #- als auch mit dem PUT-Befehl kann eine Ausgabe auf dem Drucker erfolgen. Bevor Sie einen dieser Befehle benutzen, müssen Sie einen Ausgabekanal öffnen. Über diesen Kanal erfolgt dann die Ausgabe an das angegebene Gerät. Damit die Ausgabe auf dem Drucker erfolgt, müssen Sie den Ausgabekanal für den Drucker öffnen (P:). Ein Beispiel dafür ist der folgende Befehl:

```
10 OPEN #2,8,0,"P:"
```

Danach wird jede Anweisung, die mit Hilfe des PRINT #- und PUT-Befehls Daten an Kanal 2 übergibt, eine Ausgabe auf dem Drucker zur Folge haben.
Bei Ausführung des folgenden Programmes werden zwei Zeilen auf dem Drucker ausgegeben:

```
10 REM ZWEI ZEILEN AUF DEM DRUCKER AUS
GEBEN
20 REM Ausgabekanal fuer Drucker oeffn
en
30 OPEN #3,8,0,"P:"
40 PRINT #3;"Ehrlich waehrt am laengst
en!"
50 PRINT #3;"Unrecht Gut gedeihet nich
t!"
60 CLOSE #3
```

Abgesehen davon, daß der Drucker langsamer ist, ergibt sich kein Unterschied, wenn diese Zeilen auf dem Bildschirm ausgegeben werden. Wenn Daten mit Hilfe des PRINT #-Befehls ausgegeben werden, dann ist das Ausgabeformat immer dasselbe, unabhängig davon, welches Gerät angesprochen wird. Wie sollte es anders sein, denn es wird ja nur der Kanal angegeben, über den die Ausgabe erfolgen soll. Über das damit angesprochene Gerät liegt keinerlei Information vor. Ändern Sie einfach einmal den OPEN-Befehl in Zeile 30 so ab, daß Kanal 3 für Ausgabe auf dem Bildschirm geöffnet wird (S:) und lassen Sie das Programm dann noch einmal laufen. Dieselben Zeilen werden in genau derselben Form wie vorher auf dem Drucker nun auf dem Bildschirm ausgegeben.

Gemischte Ausgabe auf Bildschirm und Drucker

Es ist möglich, innerhalb eines Programmes Daten abwechselnd auf dem Bildschirm und dem Drucker auszugeben. Wenn ein PRINT-Befehl ohne Angabe einer Kanalnummer gegeben wird, erfolgt die Ausgabe auf dem Bildschirm. Mit LPRINT wird immer der Drucker angesprochen. Diese Befehle dürfen innerhalb eines Programmes beliebig vermischt werden. Bei Verwendung von PRINT erfolgt die Ausgabe immer auf dem Bildschirm und bei Verwendung von LPRINT immer auf dem Drucker.
Aber was ist zu tun, wenn die Ausgabe mal auf dem Bildschirm und mal auf dem Drucker erfolgen soll? Wenn sie den PRINT #-Befehl verwenden, können Sie diese Entscheidung

den Anwender selbst treffen lassen. Natürlich muß diese Tatsache im Programm berücksichtigt sein. Eine Zeile muß vollständig ausgegeben worden sein, bevor zur nächsten übergegangen wird. Keine Zeile darf mehr als 38 Zeichen enthalten. Mehr als 24 Zeilen dürfen nicht auf einmal gedruckt werden, weil sonst die oberen Zeilen auf dem Bildschirm verlorengehen. Hier ein Beispiel:

```
10 DIM N$(12)
20 PRINT "Soll die Ausgabe auf dem Bil
dschirm"
25 PRINT "(B) oder auf dem Drucker (D)
erfolgen";
30 INPUT N$
40 IF N$(1,1)="B" THEN N$="S:":GOTO 70
50 IF N$(1,1)="D" THEN N$="P:":GOTO 70
60 GOTO 20:REM Andere Eingabe als B/D
65 REM Entsprechenden Kanal oeffnen
70 OPEN #4,8,0,N$
75 REM Eingabe
80 PRINT "Geben Sie einen Namen ein";
90 INPUT N$
95 REM Ausgabe auf gewuenschtem Geraet
100 PRINT #4;N$;" rueckwaerts: ";
110 FOR J=LEN(N$) TO 1 STEP -1
120 PRINT #4;N$(J,J);
130 NEXT J
140 PRINT #4:REM EOL
150 IF N$<>"ENDE" THEN 80
160 CLOSE #4:END
```

Zunächst wird der Anwender gefragt, auf welchem Gerät die Ausgabe erfolgen soll (Zeile 20 bis 60). Der entsprechende Gerätename (S: oder P:) wird einer Stringvariablen zugewiesen (Zeile 40 oder 50). Im darauffolgenden OPEN-Befehl für Kanal 4 wird diese Stringvariable benutzt, um den Gerätenamen anzugeben (Zeile 70). Die Ausgabe erfolgt dann auf dem gewünschten Gerät (Zeile 100, 120 und 140).

Die Zeilenlänge

Wenn Sie die letzte Position einer Bildschirmzeile (Spalte 39) beschreiben, erfolgt automatisch ein Zeilenvorschub. Dasselbe geschieht, sobald beim Drucker die letzte Position einer Zeile beschrieben wird. Wenn Sie mit den ATARI-Druckern 820 und 822 eine aus 40 Zeichen bestehende Zeile schreiben oder wenn Sie mit dem ATARI-Drucker 825 eine aus 80 Zeichen bestehende Zeile schreiben, werden Sie feststellen, daß der Drucker automatisch einen Zeilenvorschub erzeugt. Die rechte und linke Bildschirmbegrenzung beschränken die Zeilenlänge des Bildschirms allerdings normalerweise auf 38 Zeichen. Daher ist die Zeilenlänge des Bildschirms in der Regel kürzer als die Zeilenlänge des Druckers. Wenn eine aus 38 Zeichen bestehende Zeile ausgegeben wird, dann erfolgt auf dem Bildschirm ein Zeilenvorschub, beim Drucker hingegen nicht. Diesen Sachverhalt veranschaulicht das folgende Programm:

```
10 REM Ausgabekanaele oeffnen
20 OPEN #2,8,0,"S:"
30 OPEN #3,8,0,"P:"
35 REM Ausgabe auf Bildschirm und Druc
ker
40 FOR J=2 TO 3
```

```
45 REM Eine aus 39 Zeichen bestehende
Zeile ausgeben
50 PRINT #J;"Diese Zeile ist ein Zeich
en zu lang!!!!"
60 NEXT J
70 END
```

Auf dem Bildschirm und dem Drucker wird derselbe Text ausgegeben. Die Länge dieses Textes übersteigt die Länge einer Bildschirmzeile um ein Zeichen. Auf dem Drucker ist die Darstellung in einer Zeile hingegen möglich. Wäre der Text um ein Zeichen länger, würde auch auf dem Drucker automatisch ein Zeilenvorschub erzeugt. Beim ATARI-Drucker 825 muß eine Zeile allerdings 80 Zeichen lang sein, bevor ein Zeilenvorschub erfolgt. Dasselbe gilt auch, wenn LPRINT an Stelle von PRINT # verwendet wird.

Der Zeilenbuffer des Druckers

Der Drucker hat einen eigenen Speicher, um eine Druckzeile zwischenzuspeichern. Diesen Speicher nennt man den Zeilenbuffers des Druckers. Alle Zeichen, die mit Hilfe des PRINT #-, LPRINT- und des PUT-Befehls zum Drucker übertragen werden, gelangen zunächst in den Zeilenbuffer. Diese Zeichen werden nicht sofort, sondern erst nach der Übertragung eines EOL-Zeichens ausgegeben. Es wird dann die ganze Zeile ausgedruckt, der Zeilenbuffer gelöscht und ein Zeilenvorschub erzeugt. Wenn der Zeilenbuffer vor Empfang eines EOL-Zeichens gefüllt ist, geschieht genau dasselbe. Letzteres ist dann der Fall, wenn mehr als 39 Zeichen (beim ATARI-Drucker 825 mehr als 79 Zeichen) auf einmal ausgegeben werden. Das folgende Programm veranschaulicht das:

```
5 REM Ausgabekanal fuer den Drucker oe
ffnen
10 OPEN #3,8,0,"P:"
20 FOR J=0 TO 99
25 REM Zahlen von 0 bis 9 ausdrucken
30 PRINT #3;J-INT(J/10)*10;
35 REM Zaehlnummer jeder Zahl ausgeben
40 PRINT J
50 NEXT J
55 REM Letzte Zeile ausdrucken
60 PRINT #3
70 CLOSE #3
```

Zehnmal wird die Zahlenfolge 0123456789 ausgedruckt (Zeile 20, 30 und 50). Außerdem wird die Zählnummer jeder einzelnen Zahl ausgegeben, sobald sie an den Drucker übertragen wurde (Zeile 40). Sie werden feststellen, daß nicht jedesmal, wenn eine Zahl auf dem Bildschirm ausgegeben wird, das heißt also nach Übertragung eines Zeichens an den Drucker, auch eine Ausgabe auf dem Drucker erfolgt. Solange der Zeilenbuffer nicht gefüllt ist, wird nichts ausgedruckt. 100 Zeichen werden an den Drucker übertragen. Die letzten füllen den Zeilenbuffer nicht ganz. Aus diesem Grund wird ein EOL-Zeichen an den Drukker gesendet, damit auch diese Zeichen ausgedruckt werden (Zeile 60).
Für den Zeilenbuffer spielt es keine Rolle, woher die empfangenen Zeichen kommen. Es ist vollkommen gleichgültig, welcher Ausgabekanal und welcher Befehl (LPRINT, PRINT # oder PUT) verwendet wurde. Jedes Zeichen wird im Zeilenbuffer gespeichert und bleibt solange dort, bis ein EOL-Zeichen empfangen wurde oder der Buffer voll ist.

Ausdruck in einer bestimmten Form

Wenn einem Ausdruck eine bestimmte Form gegeben werden soll, dann wird das im Prinzip genauso gemacht wie bei der Ausgabe auf dem Bildschirm. Auch beim PRINT #- oder LPRINT-Befehl können Sie Kommas einsetzen. Sehen Sie sich noch einmal das Programm zur Berechnung des Benzinverbrauchs (Bild 4-3) an. Die Ausgabe auf dem Bildschirm erfolgte in einer ganz bestimmten Form. Es ist durchaus möglich, einen Ausdruck mit genau demselben Aussehen zu erzeugen. Dazu brauchen Sie nur die PRINT-Befehle in Zeile 50, 60, 110, 120 und 130 durch LPRINT-Befehle ersetzen. Kein Drucker kann die TAB-Funktion ausführen. Sie kann also nicht dazu verwendet werden, um einem Ausdruck eine solche Form zu geben.

```
4 REM STRINGS FUER DAS UNTERPROGRAMM
5 DIM N$(10),T$(10),BL$(40)
6 REM BL$ WIRD MIT LEERZEICHEN GEFUEL.
7 BL$(1)=" ":BL$(40)=BL$:BL$(2)=BL$
9 REM BENZINKOSTENTABELLE
10 PRINT "PREIS PRO LITER";
20 INPUT PPL
30 PRINT "DURSCHN. KM PRO LITER";
40 INPUT KPL
*42 PRINT "Soll die Ausgabe auf dem Bil
*dschirm"
*43 PRINT "(B) oder auf dem Drucker (D)
*erfolgen"
*44 INPUT T$
*46 IF T$(1,1)="D" THEN OPEN #5,8,0,"P:
*":GOTO 50
*48 OPEN #5,8,0,"S:"
*50 PRINT #5;"  KM","LITER","KOSTEN"
*60 PRINT #5;"-----","-----","------"
70 FOR KM=100 TO 1700 STEP 100
79 REM BERECHNE LITER U. RUNDE AUF/AB
80 LIT=INT(KM/KPL*10)/10
89 REM BERECHNE KOSTEN U. RUNDE AUF/AB
90 KOST=INT(PPL*LIT*100)/100
*100 NS=5:DD=0:N=KM:GOSUB 11000:PRINT #
*5;N$(1,NS),
*102 NS=5:DD=1:N=LIT:GOSUB 11000:PRINT
*#5;N$(1,NS),
*104 NS=6:DD=2:N=KOSTEN:GOSUB 11000:PRINT
* #5;N$(1,NS)
*110 NEXT KM
*120 PRINT #5
*130 PRINT #5,"KPL=";KPL,PPL;" DM PRO LIT
*ER"
140 END
```

Bild 6-1: Das Ausdrucken von Zahlenkolonnen

```
10995 REM ****************************
10996 REM * UNTERPROGRAMM RICHTET   *
10997 REM *  NACH DEZIMALPUNKT AUS  *
10998 REM ****************************
11000 T$=STR$(N)
11009 REM LOESCHE ALTE WERTE VON N$
11010 N$=BL$
11030 DP=LEN(T$)+1
11039 REM SUCHE POSIT.D.DEZIMALPUNKTES
11040 FOR J=1 TO LEN(T$)
11050 IF T$(J,J)="." THEN DP=J:J=NS
11060 NEXT J
11069 REM BERECHNE DIE LAENGE DER ZAHL
11070 NL=DP+DD
11080 N$(NS-NL+1,NS)=T$
11090 RETURN
```

Anmerkung: Gegenüber Bild 4-9 wurden nur die Zeilen geändert, die mit einem Stern versehen sind. Die Ausgabe erfolgt in der gleichen Form wie in Bild 4-10.

Bild 6-1: Das Ausdrucken von Zahlenkolonen (Fortsetzung)

Genauso wie in Kapitel 4 Zahlenkolonnen auf dem Bildschirm ausgegeben wurden, kann auch die Ausgabe auf dem Drucker erfolgen. In Bild 6-1 finden Sie eine etwas modifizierte Version des Programmes zur Berechnung des Benzinverbrauchs aus Kapitel 4 (Bild 4-9). In dieser Version kann der Anwender wählen, ob die Ausgabe auf dem Bildschirm oder auf dem Drucker erfolgen soll. Wenn "D" eingegeben wird, erfolgt die Ausgabe auf dem Drucker (Zeile 42 bis 46). Ansonsten wird der Bildschirm angesprochen (Zeile 48). An Stelle der PRINT-Befehle in Zeile 50, 60, 100, 102, 104, 120 und 130 werden jetzt PRINT #-Befehle verwendet. Sonst wurden keine Änderungen vorgenommen.

Berücksichtigung der Seitengröße

Die ATARI-Drucker berücksichtigen normalerweise nicht, daß das Druckerpapier aus einzelnen Seiten besteht, deren Länge begrenzt ist. Der Ausdruck erfolgt so, als ob das Papier von einer Rolle käme.
Es ist möglich, Programmlistings seitenweise auszudrucken. Mit Hilfe des LIST-Befehls geben Sie jeweils einen Teil des Programmes aus. Dieser Programmteil muß so groß sein, daß er auf einer Druckseite Platz findet. Die erste und die letzte auszudruckende Zeile muß so gewählt sein, daß diese Bedingung erfüllt ist.
Wenn vom Programm aus eine seitenweise Ausgabe erfolgen soll, dann ist das sehr viel weniger aufwendig. Die ausgedruckten Zeilen müssen vom Programm mitgezählt und immer wieder darauf geprüft werden, ob die maximal auf einer Druckseite darstellbare Zeilenzahl bereits erreicht ist. Wenn das der Fall ist, dann werden Leerzeilen ausgegeben, um an den Anfang der nächsten Druckseite zu gelangen. Falls es notwendig sein sollte, kann dort eine Überschrift geschrieben werden. Das folgende Programm benutzt das in Bild 6-2 dargestellte Unterprogramm, um die oben beschriebene Prüfung vorzunehmen und, falls notwendig, zur nächsten Druckseite überzugehen und dort eine Überschrift zu drucken. Geben Sie sowohl dieses Unterprogramm als auch die folgenden Zeilen ein:

```
90 GOTO 1000
994 REM --Anfang des Hauptprogrammes--
996 REM Ausgabekanal fuer Drucker oeff
nen
1000 OPEN #4,8,0,"P:"
1010 PRINT CHR$(125):REM Bildschirm lo
eschen
1020 PRINT "Positionieren Sie den Druc
ker am An-"
1025 PRINT "fang einer Seite und geben
 Sie ein,"
1030 PRINT "wieviele Zeilen gedruckt w
erden sol-"
1035 PRINT "len."
1040 INPUT Z
1045 REM Neue Seite
1050 DZ=61:GOSUB 900
1060 PRINT
1070 PRINT "Bitte warten Sie einen Aug
enblick!!"
1080 FOR J=1 TO Z
1090 FOR K=1 TO 4
1100 PRINT #4;INT(RND(0)*799)+200;"-";
INT(RND(0)*999)+1000,
1110 NEXT K
1115 REM EOL ausgeben, damit Zeile ged
ruckt wird
1120 PRINT #4
1125 REM Seite voll?
1130 DZ=DZ+1:GOSUB 900
1140 NEXT J
1150 END
```

```
897 REM ******************************
898 REM *  Unterprogramm/Neue Seite  *
899 REM ******************************
900 IF DZ<55 THEN RETURN
910 FOR J1=1 TO 66-DZ
920 LPRINT:NEXT J1
925 REM Ueberschrift drucken
930 LPRINT ,,"Ueberschrift"
940 LPRINT
945 REM Ueberschriften der Spalten dru
cken
950 LPRINT "SP. 1","SP. 2","SP. 3","SP
. 4"
955 REM Zeilenzaehler zuruecksetzen
960 DZ=3:RETURN
```

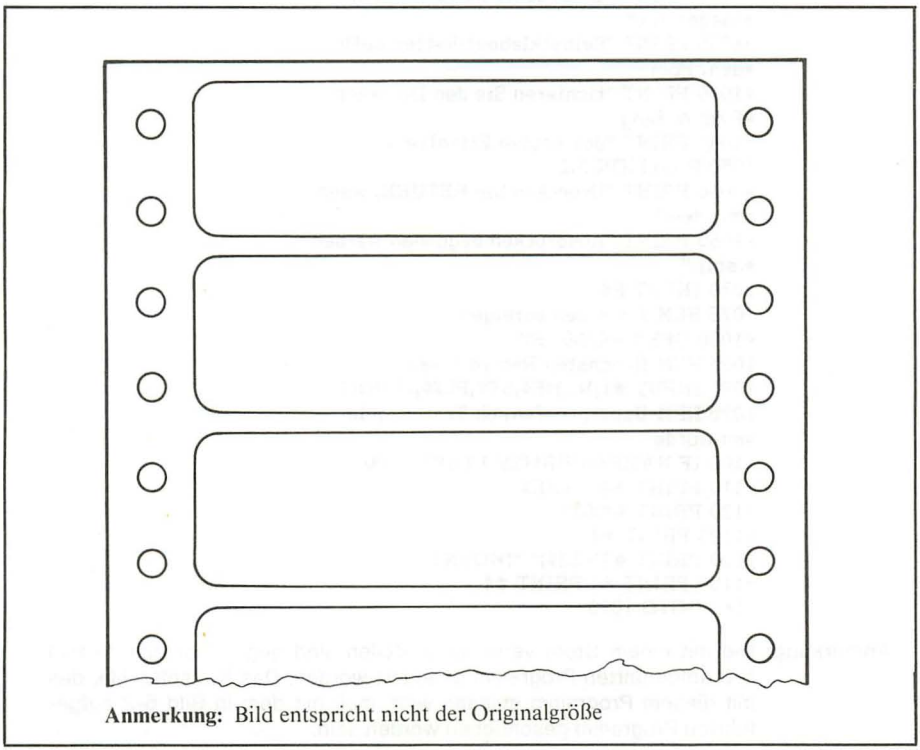
Anmerkung: Die Überschriften (Zeile 930 und 950) sind nur Beispiele.

Bild 6-2: Unterprogramm für seitenweise Ausgabe

Mit Hilfe dieses Programmes werden 4 Spalten mit 7-stelligen Zahlen ausgedruckt. Es wird davon ausgegangen, daß der Anwender den Drucker so positioniert, daß der Druckkopf in der ersten Zeile einer Seite steht. Weil DZ = 61 gesetzt wurde (Zeile 1050), erzeugt das Unterprogramm (Zeile 900) zunächst fünf Leerzeilen, bevor die Überschrift gedruckt wird. Dann werden nacheinander vier Zahlen (Zeile 1090 bis 1110) und ein EOL-Zeichen (Zeile 1120) ausgegeben, damit der Inhalt des Zeilenbuffers gedruckt wird. Danach wird der Zeilenzähler erhöht, und mit Hilfe des Unterprogrammes geprüft, ob die Seite bereits voll ist (Zeile 1130). Nachdem auf einer Seite 56 Zeilen gedruckt worden sind, wird die nächste Seite begonnen.

Ein Programmbeispiel

In Kapitel 5 haben wir ein Adressenverwaltungsprogramm erstellt, mit dessen Hilfe Namen und Adressen in einem Kassettenfile gespeichert wurden (Bild 5-2). Ein anderes Programm las dieses File wieder und brachte die Namen und Adressen auf dem Bildschirm zur Anzeige (Bild 5-3). Sehr viel praktischer wäre es, wenn diese Adressen auf Selbstklebeetiketten gedruckt werden. Solche Selbstklebeetiketten auf perforiertem Trägerpapier (Bild 6-3) sind überall leicht erhältlich.
Nur in einigen Punkten muß das Anzeigeprogramm in Bild 5-3 abgeändert werden, um damit Etiketten drucken zu können. Erstens muß in Zeile 1080 der Ausgabekanal für den



Anmerkung: Bild entspricht nicht der Originalgröße

Bild 6-3: Selbstklebeetiketten

Drucker und nicht für den Bildschirm geöffnet werden. Zweitens muß das Ausgabeformat den Etiketten entsprechen. Auf jeder 12 Zoll großen Seite des Etikettenträgers (1 Zoll = 2.54 cm) befinden sich 12 Etiketten. Die ersten Zeilen von 2 Etiketten haben einen Abstand von einem Zoll zueinander. Zum Ausdrucken einer Adresse werden vier Zeilen benötigt, wenn zwischen der Straßenbezeichnung und der Angabe des Wohnortes eine Zeile frei gelassen wird (Zeile 1125). Da der Drucker sechs Zeilen pro Zoll druckt, bleiben zwischen zwei Etiketten zwei Zeilen frei. Aus diesem Grund werden in Zeile 1150 zwei Leerzeilen gedruckt. Der Drucker kann durch Drücken von CTRL-1 nicht gestoppt werden. Also müssen Zeile 1030 bis 1040 geändert werden. In Bild 6-4 finden Sie das vollständige Programm.

```
50 DIM NAME$(20),ST$(20),PLZ$(4),WOHN$
(20),R$(20)
65 REM File oeffnen und Leerblock lese
n
70 GOSUB 3100
1000 PRINT CHR$(125);REM Bildschirm lo
eschen
*1010 PRINT "  DRUCKEN VON ADRESSENAUF
*KLEBERN"
*1020 POSITION 2,19
*1030 PRINT "Im Drucker muss sich ein T
*raeger mit"
*1035 PRINT "Selbstklebeetiketten befin
*den, Posi-"
*1040 PRINT "tionieren Sie den Druckkop
*f am Anfang
*1045 PRINT "des ersten Etikettes,
1050 POSITION 2,2
*1060 PRINT "Druecken Sie RETURN, wenn
*mit dem"
*1065 PRINT "Ausdrucken begonnen werden
* soll,"
1070 INPUT R$
1075 REM Adressen anzeigen
*1080 OPEN #4,8,0,"P:"
1085 REM Naechsten Record lesen
1090 INPUT #1,NAME$,ST$,PLZ$,WOHN$
1095 REM Ueberpruefen, ob Trailer gele
sen wurde
1100 IF NAME$=CHR$(253) THEN 1200
1110 PRINT #4;NAME$
1120 PRINT #4;ST$
*1125 PRINT #4
1130 PRINT #4;PLZ$;" ";WOHN$
*1150 PRINT #4:PRINT #4
1160 GOTO 1090
```

Anmerkung: Die mit einem Stern versehenen Zeilen sind gegenüber dem in Bild 5-3 aufgeführten Programm geändert worden. Das Kassettenfile, das mit diesem Programm gelesen wird, muß mit dem in Bild 5-2 aufgeführten Programm geschrieben worden sein.

Bild 6-4: Adressenverwaltung/Etikettendruck

```
1199 REM Trailer wurde gelesen/Program
m beenden
1200 CLOSE #1
1210 CLOSE #4
1900 END
3099 REM File fuer Programmrecorder oe
ffnen
3100 PRINT CHR$(125);"Positionieren Si
e die Kassette,"
3104 PRINT "druecken Sie RETURN und wa
rten Sie"
3106 PRINT "bitte einen Moment."
3110 OPEN #1,4,0,"C:"
3115 REM Leerblock lesen
3120 FOR J1=1 TO 128:GET #1,R:NEXT J1
3130 RETURN
```

Bild 6-4: Adressenverwaltung/Etikettendruck (Fortsetzung)

Der Zeichenvorrat der Drucker

Mit allen ATARI-Druckern können Zahlen, Satzzeichen und Groß- und Kleinbuchstaben gedruckt werden. Mit keinem Drucker können Graphikzeichen, wie ■ oder ▬ beziehungsweise invers dargestellte Zeichen gedruckt werden.
Der Zeichenvorrat der Drucker unterscheidet sich in einigen Punkten vom Zeichenvorrat des Bildschirms, weil die Zeichen unterschiedlich kodiert werden. Die Drucker verwenden zur Definition ihres Zeichenvorrats den ASCII-Kode, während bei der Zeichendarstellung auf dem Bildschirm der ATASCII-Kode verwendet wird. Für die Kodezahlen von 32 bis 127 gibt es bei den beiden Kodes kaum Unterschiede. In Teil D des Anhangs sind beide Kodes nebeneinander dargestellt. In Tabelle 6-1 finden Sie den Standardzeichenvorrat der ATARI-Drucker 820 und 822 und in Tabelle 6-2 den Zeichenvorrat des ATARI-Druckers 825. Letzterer verwendet viele der Kodezahlen, die kleiner als 32 sind, für Sonderfunktionen. Diese Sonderfunktionen werden am Ende dieses Kapitels näher betrachtet werden.

Tabelle 6-1: Standardzeichenvorrat der ATARI-Drucker 820 und 822

Kodezahl (dezimal)	Zeichen
0-31	Zwischenraum
32-95	Wie beim Bildschirm*
96	`
97-122	Wie beim Bildschirm*
123	{
124	|
125	}
126	~
127	Zwischenraum

* Die Bildschirmzeichen finden Sie in Teil D des Anhangs.

Tabelle 6-2: Zeichenvorrat des ATARI-Druckers 825

Kodezahl (dezimal)	Zeichen
0-31	Sonderfunktion (siehe Tabelle 6-4)
32-95	Wie beim Bildschirm*
96	'
97-122	Wie beim Bildschirm*
123	{
124	|
125	}
126	~
127	Kein Ausdruck

* Die Bildschirmzeichen finden Sie in Teil D des Anhangs.

Liegende Zeichen drucken (ATARI-Drucker 820)

Zusätzlich zu dem normalen Zeichensatz kann der ATARI-Drucker 820 einen großen Teil der darstellbaren Zeichen liegend drucken. Dazu müssen Sie nur beim Öffnen des Drucker-Ausgabekanals den dritten Parameter des OPEN-Befehls auf 83 setzen:

```
OPEN #4,8,83,"P:"
```

Wenn nun mit Hilfe des PRINT #- oder PUT-Befehls Zeichen über Kanal 3 zum ATARI-Drucker 820 übertragen werden, werden liegende Zeichen gedruckt. Der Zeichenvorrat der liegenden Zeichen, den Sie in Tabelle 6-3 finden, unterscheidet sich ein wenig vom normalen Zeichenvorrat. Es gibt zum Beispiel keine Kleinbuchstaben. In dieser Druckart können 29 Zeichen pro Zeile dargestellt werden.

Tabelle 6-3: Zeichenvorrat der liegenden Zeichen (ATARI-Drucker 820)

Kodezahl (Dezimal)	Zeichen	Kodezahl (Dezimal)	Zeichen	Kodezahl (Dezimal)	Zeichen	Kodezahl (Dezimal)	Zeichen	Kodezahl (Dezimal)	Zeichen
48	0	64	@	80	P	96	@	112	P
49	1	65	A	81	Q	97	A	113	Q
50	2	66	B	82	R	98	B	114	R
51	3	67	C	83	S	99	C	115	S
52	4	68	D	84	T	100	D	116	T
53	5	69	E	85	U	101	E	117	U
54	6	70	F	86	V	102	F	118	V
55	7	71	G	87	W	103	G	119	W
56	8	72	H	88	X	104	H	120	X
57	9	73	I	89	Y	105	I	121	Y
58	:	74	J	90	Z	106	J	122	Z
59	;	75	K	91	[	107	K	123	[
60	<	76	L	92	\	108	L	124	\
61	=	77	M	93	]	109	M	125	]
62	>	78	N	94	↑	110	N	126	↑
63	?	79	O	95	←	111	O	127	←

Sonderfunktionen des Druckers (ATARI-Drucker 825)

Der ATARI-Drucker 825 besitzt mehrere Sonderfunktionen. Diese Sonderfunktionen werden im folgenden beschrieben. Sie brauchen diesen Abschnitt nicht zu lesen, wenn Sie keinen ATARI-Drucker 825 haben. Folgende Sonderfunktionen gibt es:

- Papier um eine Zeile nach oben bewegen
- Papier um eine halbe Zeile nach oben bewegen
- Papier um eine Zeile nach unten bewegen
- Papier um eine halbe Zeile nach unten bewegen
- Druckkopf nach links bewegen
- Druckkopf in Bruchteilen eines Leerzeichens nach rechts oder links bewegen
- Unterstreichen
- Hoch- und tiefgestellte Zeichen drucken (Exponenten bzw. Indexdruck)
- Gedehnt drucken
- In Engschrift drucken
- Proportionalschrift
- Fettdruck
- Rechtsbündig schreiben

All diese Sonderfunktionen werden mit Hilfe von Controlzeichen aufgerufen. Die Controlzeichen sind in Tabelle 6-4 aufgeführt. Sie können auf genau dieselbe Art wie normale Zeichen an den Drucker übertragen werden, nämlich mit Hilfe des LPRINT-, PRINT #- und des PUT-Befehls. Allerdings müssen diese Zeichen ja auf irgendeine Art erzeugt werden, denn Tasten dafür gibt es nun einmal nicht. Am einfachsten geht das mit Hilfe der CHR$-Funktion. Wenn Sie zum Beispiel das Papier um eine Zeile nach oben bewegen wollen (Zeilenvorschub), dann muß ein Controlzeichen mit dem ASCII-Code 10 an den Drucker übertragen werden. Bei Ausführung des folgenden Programmes wird das Papier um 10 Zeilen nach oben bewegt:

```
5 REM Papier um 10 Zeilen nach oben be
wegen
10 FOR J=1 TO 10
20 LPRINT CHR$(10)
30 NEXT J
```

Es ist allerdings auch möglich, diese Controlzeichen direkt durch Druck bestimmter Tasten zu erzeugen, gerade so, wie Sie es auch machen, wenn Sie den Cursor bewegen oder Grafikzeichen auf den Bildschirm bringen wollen. In Tabelle 6-4 ist aufgeführt, welche Tasten Sie drücken müssen, um ein bestimmtes Controlzeichen zu erzeugen. Wenn Sie das einmal tun, werden verschiedene Grafikzeichen auf dem Bildschirm erscheinen, weil der Bildschirm den ATASCII-Kode verwendet und die Controlzeichen in diesem Kode dargestellt werden können (siehe Teil D des Anhangs). Der ATARI-Drucker 825 kann diese Zeichen allerdings nicht darstellen. Er führt bei Empfang eines solchen Zeichens die entsprechende Sonderfunktion aus. Nehmen Sie zum Beispiel CTRL-J. Wenn Sie diese Tasten drücken, erscheint ein Grafikzeichen auf dem Bildschirm. Wenn Sie dieses Zeichen jedoch an den ATARI-Drucker 825 übertragen, erfolgt dort ein Zeilenvorschub. Versuchen Sie es einmal:

```
5 REM Papier um 10 Zeilen nach oben be
wegen
10 FOR J=1 TO 10
20 LPRINT "◙"
30 NEXT J
```

Tabelle 6-4: Die Controlzeichen des ATARI-Druckers 825

Kodezahl(en) (Dezimal)	Grafik-zeichen	Taste(n)	Wirkung
10		CTRL-J	Zeilenvorschub
27 & 10		ESC\ESC & CTRL-J	Eine Zeile zurück
27 & 28		ESC\ESC & ESC\CTRL - -	Eine halbe Zeile vorwärts
27 & 30		ESC\ESC & ESC\CTRL - +	Eine halbe Zeile zurück
13		CTRL-M	Wagenrücklauf und Zeilenvorschub
15		CTRL-O	Unterstreichung/Anfang
14		CTRL-N	Unterstreichung/Ende
27 & 14		ESC\ESC & CTRL-N	Gedehnt Drucken/Anfang
27 & 15		ESC\ESC & CTRL-O	Gedehnt Drucken/Ende
27 & 19		ESC\ESC & CTRL-S	Normalschrift (10 Zeichen pro Zoll)
27 & 20		ESC\ESC & CTRL-T	Engschrift (16.7 Zeichen Pro Zoll)
27 & 17		ESC\ESC & CTRL-Q	Proportionalschrift
27 & 1		ESC\ESC & CTRL-A	Druckkopf 1 Punkt nach rechts bewegen
27 & 2		ESC\ESC & CTRL-B	Druckkopf 2 Punkte nach rechts bewegen
27 & 3		ESC\ESC & CTRL-C	Druckkopf 3 Punkte nach rechts bewegen
27 & 4		ESC\ESC & CTRL-D	Druckkopf 4 Punkte nach rechts bewegen
27 & 5		ESC\ESC & CTRL-E	Druckkopf 5 Punkte nach rechts bewegen
27 & 6		ESC\ESC & CTRL-F	Druckkopf 6 Punkte nach rechts bewegen
8 & *nn**		CTRL-H & *Taste* *	Druckkopf nn* Punkte nach links bewegen

* Mit Hilfe der auf den ASCII-Kode 8 folgenden Zeichen wird festgelegt, um wieviele Punkte der Druckkopf nach links bewegt werden soll. Die gewünschte Punktzahl nn muß der Kodezahl dieses Zeichens entsprechen. Die Taste, die das ATASCII-Zeichen mit der gewünschten Kodezahl erzeugt, können Sie Teil D des Anhangs entnehmen.

Für viele Sonderfunktionen müssen zwei Controlzeichen zum Drucker übertragen werden. Das erste Zeichen hat in diesem Fall, von einer Ausnahme abgesehen, immer den ASCII-Kode 27. Es handelt sich dabei um das sogenannte Escape-Zeichen. Zur Erzeugung dieses Zeichens benutzen Sie die CHR$-Funktion oder Sie drücken zweimal die Escape-Taste. Wenn Sie zum Beispiele das Papier um eine Zeile zurück bewegen wollen, müssen zwei Controlzeichen mit den ASCII-Kodes 27 und 10 zum Drucker übertragen werden. Im folgenden Programm wird die CHR$-Funktion dazu verwendet, um das Papier um 5 Zeilen zurück zu bewegen:

```
5 REM Papier fuenf Zeilen zurueck bewe
gen
10 FOR J=1 TO 5
20 LPRINT CHR$(27);CHR$(10)
30 NEXT J
```

Für die meisten Sonderfunktionen des ATARI-Druckers 825 werden, wie im obigen Beispiel auch, zwei Controlzeichen benötigt. Um rechtsbündig oder fett zu schreiben, muß allerdings etwas mehr Programmieraufwand getrieben werden. Ein oder zwei Controlzeichen genügen hier nicht. Dieses Problem wird später noch behandelt werden.

Controlzeichen in Programmlistings

Wundern Sie sich nicht, wenn Sie ein Programm ausdrucken lassen, in dem Controlzeichen vorkommen, und dabei seltsame Dinge passieren. Auf dem Bildschirm werden diese Controlzeichen als Grafikzeichen dargestellt. Der ATARI-Drucker 825 kann aber nicht wissen, daß nur ein Programmlisting ausgedruckt wird. Er führt die entsprechende Sonderfunktion aus und ruiniert so den ganzen Ausdruck. Nehmen wir zum Beispiel an, in einem Programm kommt die folgende Anweisung vor: "PRINT #2:"▲". Wenn Kanal 2 ein Ausgabekanal für den Drucker ist und diese Anweisung ausgeführt wird, dann erfolgt auf dem Drucker ein Zeilenvorschub. Und genau dasselbe geschieht, wenn dieses Programm mit Hilfe des Druckers aufgelistet wird. Wenn Sie zur Erzeugung der Controlzeichen die CHR$-Funktion verwenden, gibt es beim Ausdrucken des Programmlistings keine Probleme. Der Drucker führt die entsprechenden Sonderfunktionen in diesem Fall nur dann aus, wenn auch das Programm ausgeführt wird, denn erst zu diesem Zeitpunkt erfolgt die Übertragung der Controlzeichen an den Drucker. Beim Ausdrucken des Programmlistings werden ganz normale Zeichen übertragen: C, H, R, $ und so weiter.

Bewegung des Papiers nach oben oder unten

Es gibt vier Controlzeichen, mit deren Hilfe das Papier um eine halbe oder eine ganze Zeile nach oben oder unten bewegt werden kann. Dabei bewegt sich nur das Papier, nicht der Druckkopf. Das erste Controlzeichen des folgenden Programmes bewegt das Papier um eine Zeile nach unten und das zweite um eine Zeile nach oben:

```
10 LPRINT "Heute bin ich wieder";CHR$(
27);CHR$(10);" Hick ";CHR$(10);"nuecht
ern."
```

Der Ausdruck dieses Programmes sieht folgendermaßen aus:

```
                    Hick
Heute bin ich wieder      nuechtern.
```

Sie haben sicher festgestellt, daß der ATARI-Drucker 825 zum Drucken dieser Zeile drei Schritte benötigt. Nach jedem dieser Schritte erfolgt zunächst ein Wagenrücklauf. Meistens werden Sie den Zeilenvorschub (ASCII-Kode 10) jedoch dazu verwenden, das Papier sehr rasch nach oben zu bewegen. LPRINT- oder PRINT #-Befehle, die nicht mit einem Semikolon enden und bei denen kein auszudruckender Text angegeben ist, erfüllen diesen Zweck genauso. Im Gegensatz zum Zeilenvorschub wird dabei allerdings der Druckkopf bewegt. Aus diesem Grund erfolgt die Papierbewegung bei Verwendung mehrerer aufeinanderfolgender PRINT #- oder LPRINT-Befehle nicht so rasch wie bei Benutzung des Zeilenvorschubs.

Hoch- und tiefgestellte Zeichen (Exponenten- und Indexdruck)

Beim Drucken von hoch- und tiefgestellten Zeichen treten keine Probleme auf. Wenn Sie tiefgestellt drucken wollen, gehen Sie folgendermaßen vor:

1. Bewegen Sie das Papier mit Hilfe von zwei Controlzeichen mit den ASCII-Kodes 27 und 28 um eine halbe Zeile nach oben. Der Druckkopf befindet sich nun eine halbe Zeile unter der normalen Druckzeile.
2. Drucken Sie den tiefgestellten Text.
3. Bewegen Sie das Papier nun wieder um eine halbe Zeile nach unten, indem Sie die

Controlzeichen mit den ASCII-Kodes 27 und 30 zum Drucker übertragen. Der Druckkopf befindet sich nun wieder in der normalen Textzeile.

Wenn Sie hochgestellt drucken wollen, vertauschen Sie einfach Schritt 1 und 3:

1. Bewegen Sie das Papier mit Hilfe von zwei Controlzeichen mit den ASCII-Kodes 27 und 30 um eine halbe Zeile nach unten. Der Druckkopf befindet sich dann eine halbe Zeile über der normalen Druckzeile.
2. Drucken Sie den hochgestellten Text.
3. Bewegen Sie das Papier nun wieder um eine halbe Zeile nach oben, indem Sie die Controlzeichen mit den ASCII-Kodes 27 und 28 zum Drucker übertragen. Der Druckkopf befindet sich nun wieder in der normalen Textzeile.

Das folgende Programm ist ein Beispiel für den Druck von hochgestellten Zeichen:

```
10 PRINT "Die wievielte Potenz von 1,5
 soll"
15 PRINT "berechnet werden";
20 INPUT R
30 LPRINT "(1,5)";CHR$(27);CHR$(30);R;
CHR$(27);CHR$(28);"=";1.5^R
```

Wenn Sie als gewünschte Potenz 5 eingeben, sieht der Ausdruck dieses Programms folgendermaßen aus:

$(1,5)^5=7,59374994$

Unterstreichung

Wenn der ATARI-Drucker 825 ein Controlzeichen mit dem ASCII-Kode 15 empfangen hat, werden alle nachfolgend gedruckten Zeichen unterstrichen und zwar so lange, bis ein Controlzeichen mit dem ASCII-Kode 14 empfangen oder der Drucker aus- und wieder eingeschaltet wird. Das folgende Beispiel veranschaulicht diesen Sachverhalt:

```
LPRINT CHR$(15);"Krieg und Frieden";CH
R$(14);" von Leo Tolstoy"
```

Nachdem Sie diese Zeile eingegeben und RETURN gedrückt haben, wird folgendes ausgedruckt:

Krieg und Frieden von Leo Tolstoy

Zeichengröße und Zeichenlänge

In Normalschrift druckt der ATARI-Drucker 825 zehn Zeichen pro Zoll. In Engschrift werden 16.7 Zeichen pro Zoll gedruckt. Bei jeder dieser Schriftarten ist die Breite eines jeden Zeichens gleich. Es kann jedoch auch Proportionalschrift gedruckt werden. Bei dieser Schriftart wird berücksichtigt, daß ein "i" oder ein "I" weniger Platz benötigt als beispielsweise ein "M" oder ein "W" und die Zeichenbreite dementsprechend variiert. Im Durchschnitt werden dabei 14 Zeichen pro Zoll gedruckt. Bei den Zahlen von 0 bis 9 wird die Zeichenbreite allerdings nicht variiert. Dort werden dann 12.5 Zeichen pro Zoll gedruckt.
Die Auswahl der verschiedenen Schriftarten geschieht mit Hilfe von Controlzeichen. Wenn

solche Controlzeichen vom Drucker empfangen werden, ändert sich die Schriftart mit dem Ausdruck des nächsten Zeichens. Allerdings ändert sich die Schriftart nicht sofort, wenn in einer Zeile bisher in Normalschrift geschrieben und dann auf Proportional- oder Engschrift umgeschaltet wird. Erst mit dem Beginn der nächsten Druckzeile wird die Schriftart gewechselt. Genauso ist es auch, wenn in einer Zeile bisher in Proportional- oder Engschrift geschrieben und dann auf Normalschrift umgeschaltet wird. Hieraus ergibt sich, daß nur Proportional- und Engschrift in einer Druckzeile gemischt werden können. Eine Mischung von Proportional- oder Engschrift mit Normalschrift ist nicht möglich.
In einer einmal gewählten Schriftart wird solange gedruckt, bis mit Hilfe eines entsprechenden Controlzeichens auf eine andere Schriftart umgeschaltet wird. Nach dem Einschalten des Druckers wird in Normalschrift gedruckt. Zusätzlich kann jedes Zeichen gedehnt gedruckt werden. Es ist dann doppelt so breit wie normal. Diese Funktion wird durch Übertragung der Controlzeichen mit den ASCII-Kodes 27 und 14 angewählt. Danach werden solange alle Zeichen gedehnt gedruckt, bis die Controlzeichen mit den ASCII-Kodes 27 und 15 empfangen werden oder ein Wagenrücklauf erfolgt.
Mit Hilfe des folgenden Programmes werden alle Schriftarten dargestellt (welche Tasten Sie zur Erzeugung der Graphikzeichen betätigen müssen, können Sie Tabelle 6-4 entnehmen):

```
10 DIM NORM$(2),ENG$(2),PROP$(2),BR$(2
)
15 Controlzeichen in Variablen speiche
rn
20 NORM$="[ESC][+]":REM Normalschrift
30 ENG$="[ESC][●]":REM Engschrift
40 PROP$="[ESC][┌]":REM Proportionalschrift
95 REM Ausgabekanal fuer Drucker oeffn
en
100 OPEN #3,8,0,"P:"
110 PRINT #3;NORM$;"Normalschrift"
120 GOSUB 1000
130 PRINT #3;ENG$;"Engschrift"
140 GOSUB 1000
150 PRINT #3;PROP$;"Proportionalschrif
t"
160 GOSUB 1000
200 END
999 REM Zeichenvorat ausdrucken
1000 BR$="":REM Normale Breite
1010 FOR L=1 TO 2
1020 FOR K=32 TO 96 STEP 32
1025 REM Zeichenbreite waehlen
1030 PRINT #3;BR$;
1035 REM Die naechsten 32 Zeichen druc
ken
1040 FOR J=1 TO 31
1050 PRINT #3;CHR$(J+K);
1060 NEXT J
1065 REM Zeilenbuffer ausdrucken
1070 PRINT #3
1080 NEXT K
1085 REM Doppelte Breite
1090 BR$="[ESC][_]"
1095 REM Und noch einmal den Zeichenvo
rat drucken
1100 NEXT L
1110 PRINT #3:REM Leerzeile
1120 RETURN
```

Tabelle 6-5: Zusammenfassung der verschiedenen Zeichenbreiten beim ATARI-Drucker 825 (*)

Schriftart	Zeichenbreite
Normal	10 Punkte
Eng	9 Punkte
Proportional	6 – 18 Punkte

* Bei gedehnter Zeichendarstellung verdoppeln sich die angegebenen Werte.

Tabelle 6-6: Zeichenbreite bei der Darstellung in Proportionalschrift (ATARI-Drucker 825)

Kodezahl (Dezimal)	Zeichen	Zeichenbreite in Punkten	Kodezahl (Dezimal)	Zeichen	Zeichenbreite in Punkten	Kodezahl (Dezimal)	Zeichen	Zeichenbreite in Punkten
32	Leerzeichen	7	64	@	14	96	`	7
33	!	7	65	A	16	97	a	12
34	"	10	66	B	15	98	b	12
35	#	15	67	C	14	99	c	10
36	$	12	68	D	16	100	d	12
37	%	16	69	E	14	101	e	12
38	&	14	70	F	14	102	f	10
39	'	7	71	G	16	103	g	12
40	(	7	72	H	16	104	h	12
41	)	7	73	I	10	105	i	8
42	*	12	74	J	14	106	j	6
43	+	12	75	K	16	107	k	12
44	,	7	76	L	14	108	l	8
45	-	12	77	M	18	109	m	16
46	.	7	78	N	16	110	n	12
47	/	12	79	O	16	111	o	12
48	0	12	80	P	14	112	p	12
49	1	12	81	Q	14	113	q	12
50	2	12	82	R	15	114	r	10
51	3	12	83	S	12	115	s	12
52	4	12	84	T	14	116	t	10
53	5	12	85	U	16	117	u	12
54	6	12	86	V	16	118	v	12
55	7	12	87	W	18	119	w	16
56	8	12	88	X	16	120	x	12
57	9	12	89	Y	16	121	y	12
58	:	7	90	Z	10	122	z	10
59	;	7	91	[	12	123	{	10
60	<	12	92	\	12	124	|	7
61	=	12	93	]	12	125	}	10
62	>	12	94	^	12	126	~	12
63	?	12	95	—	12	127	entfällt	0

* Bei gedehnter Zeichendarstellung verdoppeln sich die angegebenen Werte.

Weil beim ATARI-Drucker 825 jedes Druckzeichen aus einzelnen Punkten besteht, wird die Zeichenbreite in Punkten angegeben. Die Zeichenbreite bewegt sich zwischen 6 Punk-

ten (normale Breite) für das kleinste und 36 Punkten (doppelte Breite) für das größte Zeichen, das in Proportionalschrift dargestellt wird. Bei der Darstellung in Normal- und Engschrift liegt die Zeichenbreite zwischen diesen beiden Extremen, normalerweise bei 10 beziehungsweise 9 Punkten. In Tabelle 6-5 sind die verschiedenen Zeichenbreiten zusammengefaßt. In Tabelle 6-6 finden Sie die Zeichenbreite für jedes einzelne Zeichen bei der Darstellung in Proportionalschrift.
Die maximale Zeilenlänge des ATARI-Druckers 825 beträgt 8 Zoll. Beachten Sie, daß automatisch ein Wagenrücklauf erfolgt, wenn eine Zeile voll ist. In Normalschrift können in einer Zeile 80 Zeichen, in Engschrift 132 Zeichen gedruckt werden. Da die Zeichenbreite bei Proportionalschrift variiert, muß man hier nicht die Zeichen, sondern die Punkte zählen. 1200 Punkte können in einer Zeile gedruckt werden. Der Drucker sieht die Zeile allerdings bereits als gefüllt an, wenn erst 1185 Punkte gedruckt wurden. Es wird dann noch ein Zeichen, gleich welcher Breite, gedruckt und automatisch ein Wagenrücklauf durchgeführt. Wenn dieser automatische Wagenrücklauf nicht erfolgen soll, dürfen nicht mehr als 1185 Punkte in einer Zeile gedruckt werden. Wenn in doppelter Breite gedruckt wird, reduzieren sich alle oben angegebenen Werte auf die Hälfte. Das heißt, in einer Zeile können bei Darstellung in doppelter Breite in Normalschrift 40 und in Engschrift 66 Zeichen beziehungsweise 600 Punkte gedruckt werden, bevor ein automatischer Wagenrücklauf erfolgt.

Den Druckkopf in Bruchteilen eines Leerzeichens vor und zurück bewegen

Normalerweise können Sie den Druckkopf nur nach rechts bewegen und zwar mit Hilfe des Leerzeichens (ASCII-Kode 32). Dabei ist ganz genau festgelegt, wie weit sich der Druckkopf bewegt. In Normalschrift belegt ein Leerzeichen 10 Punkte. Zusätzlich gibt es nun noch sechs Sonderfunktionen, die den Druckkopf nur um einen Bruchteil des normalen Leerzeichens, nämlich ein bis sechs Punkte, nach rechts bewegen. Zur Ausführung dieser Funktionen müssen zwei Controlzeichen zum Drucker übertragen werden. Das erste hat den ASCII-Kode 27 und das zweite gibt an, um wieviel Punkte der Druckkopf bewegt werden soll (ASCII-Kode 1 bis 6). Am besten und am einfachsten ist es, wenn Sie zur Erzeugung dieser Controlzeichen die CHR$-Funktion verwenden, so wie es auch im folgenden Programm gemacht wurde:

```
5 REM Ausgabekanal fuer den Drucker oe
ffnen
10 OPEN #5,8,0,"P:"
15 REM Wortzwischenraum 1 bis 6 Punkte
20 FOR J=1 TO 6
30 PRINT #5;"AHA";CHR$(27);CHR$(J);
40 NEXT J
50 PRINT #5;"SOSO"
```

Der Ausdruck dieses Programmes sieht folgendermaßen aus:

```
AHA AHA AHA AHA AHA AHA SOSO
```

Diese Sonderfunktionen finden vor allem dann Verwendung, wenn rechtsbündig oder fett gedruckt werden soll. Aber darauf wird noch näher eingegangen. Der Druckkopf des ATARI-Druckers 825 kann darüberhinaus auch nach links bewegt werden. Allerdings müssen Sie dabei immer angeben, um wieviel Punkte sich der Druckkopf bewegen soll, denn ein "Leerzeichen rückwärts" gibt es nicht. Wenn Ihnen nur die Anzahl der zu überspringenden Zeichen bekannt ist, dann berechnen Sie

zuerst mit Hilfe von Tabelle 6-5 die entsprechende Punktzahl. Dann wird ein Controlzeichen mit dem ASCII-Kode 8 zum Drucker übertragen. Der ASCII-Kode des darauf folgenden Zeichens gibt an, um wieviel Punkte der Druckkopf nach links bewegt wird. Die Kodezahl dieses Zeichens muß zwischen 0 und 127 liegen. Das folgende Programm veranschaulicht diesen Sachverhalt:

```
5 REM Ausgabekanal fuer den Drucker oe
ffnen
10 OPEN #5,8,0,"P:"
20 PRINT #5;"Mit diesem Dingsda";
25 REM Die letzten sieben Zeichen uebe
rschreiben
30 PRINT #5;CHR$(8);CHR$(70);
40 PRINT #5;"------- Drucker kann man
auch durchstreichen!!!"
```

In Zeile 30 wird zunächst das Controlzeichen mit dem ASCII-Kode 8 und dann die Anzahl der Punkte, um die der Druckkopf nach links bewegt werden soll, zum Drucker übertragen. In diesem Fall sind das 70 Punkte, was in Normalschrift sieben Zeichen entspricht. Gleich darauf werden 7 Bindestriche gedruckt. Da sich der Druckkopf in der Zwischenzeit bewegt hat, werden diese Bindestriche über die letzten sieben Zeichen gedruckt.

```
Mit diesem Dingsda Drucker kann man auch durchstreichen!!!
```

Sicher haben Sie bemerkt, daß der Druckkopf nicht wirklich um das angegebene Stück nach links wandert. Stattdessen wird er zunächst ganz nach links und dann wieder soweit nach rechts bewegt, daß der gewünschte Abstand zum Ausgangspunkt vorhanden ist. Betrachtet man nur den Anfangs- und den Endzustand, dann sieht das Ganze so aus, als ob der Druckkopf um das angegebene Stück nach links gerückt wäre.
An Stelle der CHR$-Funktion kann auch direkt das Zeichen mit dem entsprechenden ASCII-Kode eingesetzt werden. Im vorliegenden Fall müßte CHR$(70) durch "F" ersetzt werden, denn "F" hat den ASCII-Kode 70. Zeile 30 würde dann so aussehen:

```
30 PRINT #5;CHR$(8);"F"
```

Nachdem das letzte Zeichen einer Zeile zum Drucker übertragen wurde, kann der Druckkopf nicht mehr nach links bewegt werden, denn bei Empfang des ersten Controlzeichens würde die Zeile sofort ausgedruckt.

Fettdruck

Um bestimmte Textteile besonders hervorzuheben, wird häufig fett gedruckt. Das ist zum Beispiel dadurch zu erreichen, indem man diese Textteile mehrmals übereinander druckt, wie es auch im folgenden Beispiel gemacht wurde:

```
5 REM Ausgabekanal fuer den Drucker oe
ffnen
10 OPEN #5,8,0,"P:"
20 PRINT #5;"Diese Zeile wurde einmal
gedruckt,"
30 FOR J=1 TO 2
40 PRINT #5;"Zur besonderen Hervorhebu
ng wurde diese Zeile zweimal gedruckt,
"
```

```
45 REM Eine Zeile zurueck
50 PRINT #5;CHR$(27);CHR$(10);
60 NEXT J
65 REM Den letzten Ruecktransport ruec
kgaengig machen
70 PRINT #5
80 END
```

In diesem Programm wird eine FOR-NEXT-Schleife dazu verwendet, eine Zeile mehrmals übereinander zu drucken (Zeile 40 bis 60). Nach dem Ausdruck dieser Zeile erfolgt auf dem Drucker automatisch ein Zeilenvorschub (Zeile 40). Dieser Zeilenvorschub muß vom Programm wieder rückgängig gemacht werden. Wenn die FOR-NEXT-Schleife endet, wurde das Papier gerade um eine Zeile zurück transportiert. Dieser Rücktransport muß wieder rückgängig gemacht werden. Deshalb erfolgt vor dem Programmende noch einmal ein Zeilenvorschub (Zeile 70).
Das Ergebnis sieht folgendermaßen aus:

```
Diese Zeile wurde einmal gedruckt.
Zur besonderen Hervorhebung wurde diese Zeile zweimal gedruckt.
```

Dieser Effekt kann noch gesteigert werden, wenn der Druckkopf beim zweiten Druckvorgang gegenüber dem ersten um einen Punkt verschoben wird. Erweitern Sie das obige Programm folgendermaßen:

```
80 FOR J=1 TO 2
90 PRINT #5;"Zur besonders starken Her
vorhebung wurde diese Zeile fett gedru
ckt."
95 REM Eine Zeile zurueck
100 PRINT #5;CHR$(27);CHR$(10)
105 REM Druckkopf um einen Punkt nach
rechts bewegen
110 PRINT #5;CHR$(27);CHR$(1)
120 NEXT J
125 REM Den letzten Ruecktransport rue
ckgaengig machen
130 PRINT #5
140 END
```

Der Ausdruck dieses Programmes sieht folgendermaßen aus:

```
Diese Zeile wurde einmal gedruckt.
Zur besonderen Hervorhebung wurde diese Zeile zweimal gedruckt.
Zur besonders starken Hervorhebung wurde diese Zeile fett gedruckt.
```

Rechtsbündig drucken

Das rechtsbündige Drucken wird erst dadurch ermöglicht, daß der Druckkopf in Bruchteilen eines Leerzeichens nach rechts oder links bewegt werden kann. Zunächst müssen Sie feststellen, wie groß der Abstand (in Punkten) des letzten Zeichens einer Zeile bis zum rechten Rand ist. Diese Punkte werden dann innerhalb der Zeile verteilt. Sie fügen also ab und zu zwischen zwei Zeichen einen Punkt Leerraum ein. Innerhalb einer Zeile müssen diese Leerräume gleichmäßig verteilt werden, damit das Druckbild nicht unausgewogen wirkt. Ein gutes Programm muß feststellen können, wo diese Leerräume am besten eingefügt, beziehungsweise wo sie als störend empfunden werden.
Ein solches Programm zu schreiben ist nicht leicht, selbst dann nicht, wenn nur eine Mini-

malausführung erstellt wird. Ein BASIC-Programm wird nicht sehr schnell sein. Als Übung ist es aber trotzdem ganz nützlich, einmal ein solches Programm zu entwickeln. Optimal ist es allerdings, wenn es in Assembler geschrieben wird.

7

DIE ATARI-DISKETTENSTATION 810

Die Diskettenstation ist ein sehr wichtiger Bestandteil eines Computersystems. Mit ihrer Hilfe kann sehr schnell auf sehr große Datenmengen zugegriffen werden. Bei der ATARI-Diskettenstation 810 können auf einer einzigen Diskette ungefähr 90000 Zeichen gespeichert werden. Das ist fast doppelt soviel wie die 48K eines voll ausgebauten ATARI-Computers. Außerdem sind diese Daten dauerhaft gespeichert. Demgegenüber gehen alle im Hauptspeicher des Computers befindlichen Daten beim Ausschalten verloren.

Die verschiedenen Diskettenarten

Die Informationsspeicherung erfolgt bei Disketten magnetisch wie bei einem Tonbandgerät. Der größte Unterschied zum Tonbandgerät besteht darin, daß eine Diskette rund ist wie eine Schallplatte. Und genauso wie eine Schallplatte sich auf dem Plattenteller dreht, dreht sich auch die Diskette in der Diskettenstation. Mit Hilfe eines Magnetkopfes kann die Diskette gelesen oder beschrieben werden. Diesen Magnetkopf können Sie mit der Nadel eines Plattenspielers vergleichen. An jedem beliebigen Punkt einer Schallplatte können Sie die Nadel aufsetzen. Es macht keinen Unterschied, ob Sie ein Stück vom Ende der Platte oder aus der Mitte oder vom Anfang hören wollen. Genauso kann der Computer den Schreib-Lese-Kopf der Diskettenstation auf jeder beliebigen Stelle der Diskette positionieren. Man nennt das wahlfreien Zugriff (Random Access). Die Diskette ist also ein Speichermedium für wahlfreien Zugriff.
Man unterscheidet drei Arten von Disketten: "Hard Disks", "Winchester Disks" und "Floppy Disks". Zur Zeit werden für Mikrocomputer wie den ATARI 400/800 normalerweise Floppy Disks verwendet. Im folgenden werden alle drei Diskettenarten beschrieben.

Hard Disks

Die Disketten für diese Laufwerke werden aus einem massiven Material wie Aluminium hergestellt und mit einer magnetischen Substanz beschichtet. Auf ihnen können 5 bis 10 Millionen Zeichen gespeichert werden. Meistens befinden sich diese Disketten in einem eigenen Gehäuse, mit dem sie zusammen in das Laufwerk eingelegt werden (Bild 7-1). Aufgrund dieser Konstruktion können die Disketten bei Hard Disks gewechselt werden. Die Disketten kosten ungefähr 350 DM und die Laufwerke 7500 bis 25000 DM.

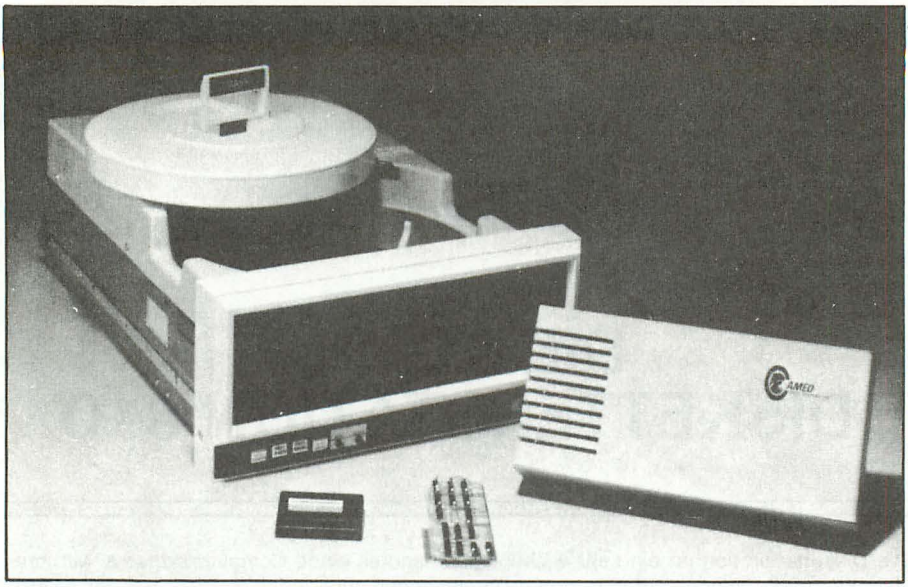


Photo courtesy of Cameo Electronics

Bild 7-1: Hard Disk mit auswechselbarer Diskette

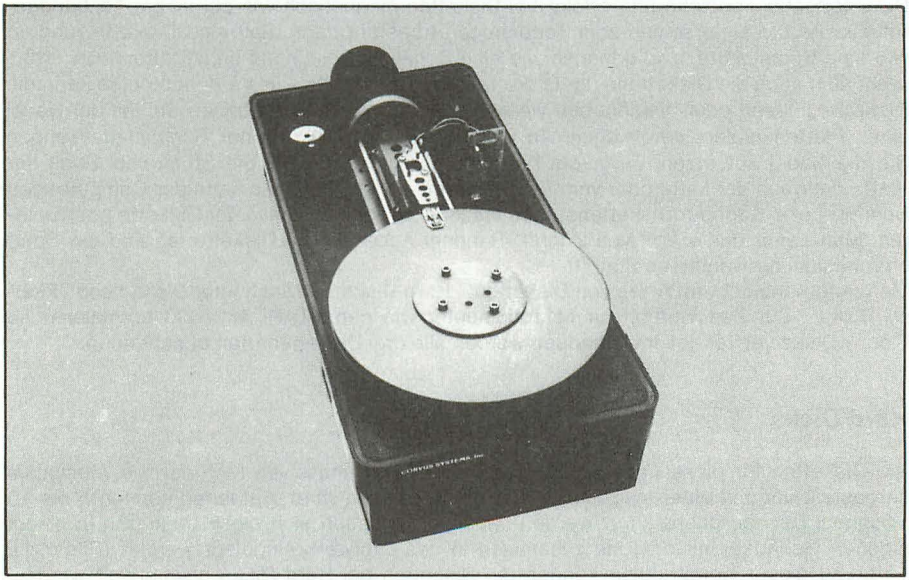


Photo courtesy of Corvus Systems, Inc.

Bild 7-2: Winchester Disk (Gehäuse geöffnet)

Winchester Disks

Durch eine besondere Technik ist es möglich, bei dieser Laufwerksart sechs bis zehnmal soviel Daten auf einer Diskette zu speichern wie bei Hard Disks. Winchester Disks (Bild 7-2) sind allerdings extrem empfindlich gegenüber Schmutz und Staub. Sogar Zigarettenrauch kann zum Ausfall führen. Um Umwelteinflüsse so weit wie möglich abzuschirmen, werden die Disketten meistens in den Laufwerken eingeschlossen. Es ist nicht möglich, sie ohne weiteres zu wechseln. Winchester Disks kosten 6500 bis 20000 DM.

Floppy Disks

Bei dieser Laufwerksart besteht die Diskette aus Vinyl. Sie ist flexibel und ziemlich empfindlich. Deshalb befindet sie sich in einer Kunststoffhülle, die sie vor Beschädigungen schützen soll. Entfernen Sie diese Umhüllung niemals. In Bild 7-3 sehen Sie, wie es innerhalb der Schutzhülle aussieht.
In der Mitte der Hülle befindet sich ein Loch, durch das die Diskette vom Laufwerk festgehalten und innerhalb der Hülle gedreht werden kann.
Diese Disketten gibt es in zwei Größen: mit 8 Zoll oder 5 1/4 Zoll Durchmesser. Die kleineren nennt man auch Mini-Disks beziehungsweise Mini-Disketten. Diese Disketten werden auch für das ATARI-Diskettenlaufwerk 810 verwendet. Dort werden auf einer Diskette 92160 Zeichen gespeichert.

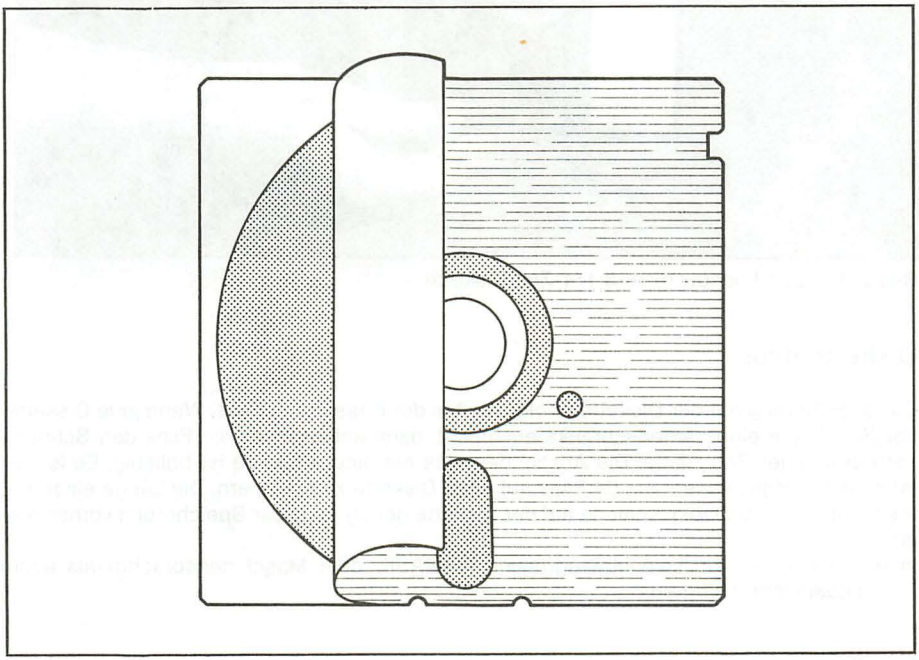


Bild 7-3: So sieht es innerhalb der Schutzhülle aus

Der Schreibschutz

Jede Diskette hat seitlich eine kleine Aussparung. Mit Hilfe dieser Aussparung kann die Diskette vor Beschreiben geschützt werden. Wenn sie nicht vorhanden ist, kann die Diskette nicht beschrieben werden.
Manche Disketten haben diese Aussparung von vorneherein nicht. Sie können niemals versehentlich beschrieben werden. Sie selbst können eine Diskette dadurch schützen, indem Sie diese Aussparung mit einem speziellen Aufkleber oder irgendeinem anderen Stück undurchsichtigen Klebeband überkleben (Bild 7-4).

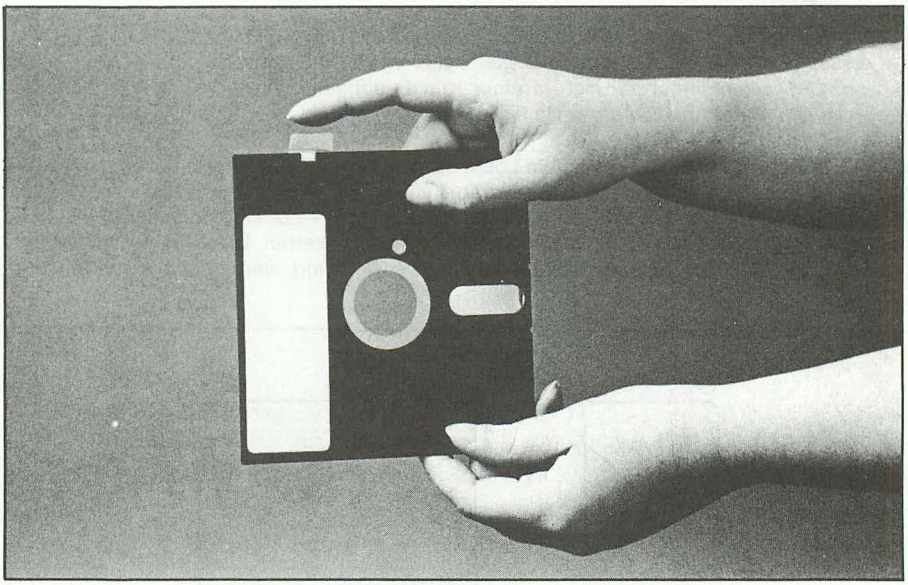


Bild 7-4: Schreibschutz einer 5 1/4-Zoll-Diskette

Diskettenfiles

Die Speicherung auf der Diskette erfolgt wie bei der Kassette in Files. Wenn jede Diskette der Schublade eines Aktenschranks entspricht, dann entsprechen die Files den Schnellheftern in einer Schublade. Die Anzahl der Files auf einer Diskette ist beliebig. Es ist jedoch nicht möglich, mehr als 64 Files auf einer Diskette zu speichern. Die Länge eines Files ist ebenfalls beliebig, solange auf der Diskette genug Platz zur Speicherung vorhanden ist.
In einem File können sowohl Programme (in BASIC oder Maschinensprache) als auch Daten gespeichert werden.

Der Filename

Jedes File bekommt einen Namen, damit es von anderen Files unterschieden werden kann. Dieser Name darf maximal acht Zeichen lang sein. Zur Namensgebung können Großbuchstaben oder Zahlen verwendet werden. Das erste Zeichen eines Filenamens muß allerdings immer ein Großbuchstabe sein. Sonderzeichen wie $, § oder #, Leerzeichen oder Satzzeichen dürfen nicht verwendet werden.

Die Erweiterung des Filenamens

Jeder Filename kann um ein, zwei oder drei Zeichen erweitert werden. Dieser Zusatz ist eine beliebige Kombination von Buchstaben und Zahlen. Im Gegensatz zum Filenamen selbst darf hier auch das erste Zeichen eine Zahl sein. Falls Sie einem Filenamen einen Zusatz geben wollen, schreiben Sie hinter den Filenamen zuerst einen Punkt und dann den gewünschten Zusatz. Wenn zum Beispiel der Filename TEST und der Zusatz TXT lauten würde, dann sähe der vollständige Filename so aus: TEST.TXT.
Diesen Zusatz verwendet man oft dazu, um festzuhalten, um welchen Filetyp es sich handelt. In Tabelle 7-1 finden Sie eine Zusammenstellung von allgemein gebräuchlichen Zusätzen und den entsprechenden Filetypen. Den Zusatz .SYS sollten Sie allerdings nicht verwenden, weil er für Systemfiles reserviert ist.

Tabelle 7-1: Allgemein gebräuchliche Zusätze von Filenamen

Zusatz	Filetyp
SYS	Systemfile. Ein Systemfile enthält ein Systemprogramm wie das Disk Operating System (DOS) oder Programmiersprachen wie zum Beispiel Microsoft BASIC.
BAS	File mit einem BASIC-Programm, das mit Hilfe des SAVE-Befehls gespeichert wurde.
LST	File mit einem BASIC-Programm, das mit Hilfe des LIST-Befehls gespeichert wurde.
ASM	File mit einem Programm, das in Assembler geschrieben wurde.
TXT	Textfiles.
DAT	Datenfile. Manchmal wird auch DTA verwendet.
OBJ	Objektkodefile. File mit einem Programm in Maschinensprache (assembliertes ASM-File).
BAK	Backupfile. Kopie eines bereits vorhandenen Files. Wird erst dann benutzt, wenn das Originalfile versehentlich gelöscht wurde.
TMP	File, das nur für eine kurze Zeit (temporär) benötigt wird.

Das Inhaltsverzeichnis der Diskette (DIRECTORY)

Ein kleiner Teil von jeder Diskette ist für das Inhaltsverzeichnis reserviert. In diesem Inhaltsverzeichnis wird der Name von jedem auf der Diskette befindlichen File, seine Größe und seine genaue Lage auf der Diskette festgehalten. Wenn Sie einen Filenamen eingeben, sieht der Computer in diesem Inhaltsverzeichnis nach, ob der Name schon existiert und, wenn ja, wo sich das File befindet.

Das Disk Operating System (DOS)

Das Disk Operating System (DOS) ist ein Diskettenbetriebsprogramm. Mit seiner Hilfe kann die Diskettenstation überhaupt erst benutzt werden. Auch wenn die Diskettenstation in einem BASIC-Programm angesprochen wird, dann wird die gewünschte Operation vom Disk Operating System ausgeführt und das Ergebnis an das BASIC-Programm übergeben. Das Disk Operating System selbst ist in Maschinensprache geschrieben.

Die verschiedenen Versionen des Disk Operating System

Zur Zeit gibt es zwei verschiedene Versionen des Disk Operating Systems. Als erste Version wurde DOS 1.0 von ATARI herausgegeben. Bis Ende 1981 wurden alle ATARI-Diskettenstationen 810 mit dieser DOS-Version ausgeliefert. Inzwischen wurde DOS 1.0 durch DOS 2.0S ersetzt. Wenn Sie wissen wollen, mit welcher Version Sie arbeiten, sehen Sie einfach auf dem Etikett der Diskette nach. Die besonderen Eigenschaften der beiden Ver-

sionen werden später noch behandelt.
In diesem Kapitel wird sowohl DOS 1.0 als auch DOS 2.0S beschrieben. Die Beschreibung von DOS 2.0S entspricht dem Stand von Dezember 1981.
Die beiden Versionen des Disk Operating System sind zwar sehr ähnlich aber nicht identisch. Mit DOS 2.0S können Disketten gelesen werden, die mit DOS 1.0 beschrieben wurden, aber bei dem Versuch, mit DOS 1.0 eine mit DOS 2.0S beschriebene Diskette zu lesen, können ab und zu Fehler auftreten. Schreiben Sie deshalb immer nur mit ein und derselben DOS-Version auf eine Diskette. Wenn Sie das nicht tun, könnten Daten auf dieser Diskette verlorengehen. Falls Sie doch einmal das DOS wechseln sollten, versehen Sie die entsprechende Diskette zur Sicherheit vorher mit einem Schreibschutz.

Die beiden Teile des Disk Operating System

Das Disk Operating System besteht aus zwei Teilen. Mit Hilfe des einen Teils können Sie BASIC-Programme oder Datenfiles laden oder abspeichern. Der andere Teil besteht aus verschiedenen Hilfsprogrammen zum Betrieb der Diskettenstation. Unter anderem können Sie damit auch Datenfiles oder Programme, die in Maschinensprache geschrieben wurden, laden oder abspeichern.
In DOS 1.0 sind diese beiden Teile in einem Programm zusammengefaßt. Sie sind in einem File namens DOS.SYS auf der Diskette gespeichert. In DOS 2.0S gibt es stattdessen zwei Programme, die in zwei Files gespeichert sind. Diese Aufteilung wurde vorgenommen, um den Hauptspeicher des Computers besser ausnutzen zu können. Der erste Teil (DOS.SYS) wird immer benötigt, wenn Sie auf die Diskette zugreifen, aber der zweite Teil (DUP.SYS) wird nur dann gebraucht, wenn eines der Hilfsprogramme ausgeführt werden soll. Durch die Aufteilung in zwei Programme wird der Speicherplatz frei, der sonst durch die Hilfsprogramme belegt würde.

Der Datenfluß zwischen Computer und Diskettenstation

Der Datenfluß zwischen dem Computer und der Diskettenstation wird vom Disk Operating System überwacht und gesteuert. Die Datenübertragung erfolgt in Blöcken zu jeweils 128 Bytes. Ein Teil des Hauptspeichers wird für die Zwischenspeicherung eines solchen Blocks reserviert. Diesen Speicherbereich nennt man Diskettenbuffer. Da maximal vier Diskettenstation gleichzeitig an den ATARI-Computer angeschlossen werden können, legt das Disk Operating System vier Diskettenbuffer an.
Wenn während eines Ladevorganges Programmteile oder Daten vom Computer benötigt werden, dann versucht das Disk Operating System zunächst, diesen Bedarf mit Hilfe des Diskettenbuffers zu decken. Wenn das nicht möglich ist, wird der nächste Block von der Diskette geladen.
Bei der Speicherung von Programmen oder Daten läuft derselbe Vorgang mit entgegengesetztem Vorzeichen ab. Die Programme oder Daten werden zunächst im Diskettenbuffer zwischengespeichert. Erst wenn der Buffer gefüllt ist, wird ein Block auf die Diskette geschrieben.

Inbetriebnahme der Diskettenstation

Bevor Sie mit der Diskettenstation arbeiten können, muß sich das Disk Operating System im Speicher befinden. Wenn Sie genug Zeit hätten, könnten Sie es mit Hilfe der Tastatur eingeben. Es gibt allerdings einen wesentlich einfacheren Weg: das "Booten". Bei diesem Bootvorgang wird das Disk Operating System von der Diskette in den Hauptspeicher des Computers geladen.

Der Bootvorgang

In Kapitel 2 wurde der Bootvorgang ausführlich beschrieben. Wenn Sie Ihr Computersystem entsprechend dieser Beschreibung in Betrieb nehmen, wird das Disk Operating System sozusagen nebenbei geladen. Hier noch einmal eine Zusammenfassung der wichtigsten Schritte:

- Schalten Sie die Diskettenstation Nr. 1 ein. Wenn Sie mehrere Diskettenstationen besitzen und nicht genau wissen, welches Laufwerk 1 ist, sehen Sie sich die Schalter auf der Rückseite der Diskettenstation an. Laufwerk 1 ist diejenige, bei der sich sowohl der schwarze als auch der weiße Teil des Schalters in seiner linken Schaltposition befindet (Bild 2-5).
- Legen Sie eine Diskette, auf der sich eine Kopie des Disk Operating System (Filename DOS.SYS) befindet, in Laufwerk 1 ein. Die Disketten, die ein Etikett mit der Aufschrift "Disk File Manager Master Copy" (DOS 1.0) oder "Disk File Manager II Master Copy" (DOS 2.0S) tragen, enthalten auf jeden Fall ein DOS.
- Schalten Sie Ihren ATARI-Computer aus und wieder ein. Daraufhin wird Laufwerk 1 anlaufen und das DOS wird in den Hauptspeicher des Computers geladen.

Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, können Sie den Bootvorgang auch akustisch mitverfolgen. Nach etwa 20 Sekunden wird wieder READY auf dem Bildschirm erscheinen.
Es sieht zunächst so aus, als ob gar nichts passiert wäre, denn es gibt eigentlich nichts, was darauf hindeutet, daß das Disk Operating System während dieser 20 Sekunden geladen wurde. Das es dennoch so ist, werden Sie feststellen, wenn Sie versuchen, auf die Diskette zuzugreifen. Sie werden sehen, daß es funktioniert.
Das DOS kann von jeder Diskette gebootet werden, auf der sich ein File namens DOS.SYS befindet. Während des Bootvorganges wird das in Maschinensprache geschriebene Disk Operating System von diesem File zum Computer übertragen. Bei DOS 1.0 werden beide Teile des Disk Operating System geladen, bei DOS 2.0S nur der erste Teil. Der zweite Teil mit den Hilfsprogrammen für den Betrieb der Diskettenstation wird vorläufig noch nicht geladen. In jedem Fall bleibt der Inhalt von DOS.SYS bis zum Ausschalten des Computers gespeichert.

Die Diskettenhilfsprogramme

Nach dem Booten des DOS können alle für den Betrieb der Diskettenstation vorgesehenen Befehle benutzt werden. Einer davon ist der DOS-Befehl. Geben Sie ihn einmal ein und drücken Sie RETURN:

```
DOS
```

Daraufhin wird das DOS-Menue (Bild 7-5 oder 7-6) auf dem Bildschirm erscheinen. Bei Verwendung von DOS 1.0 erscheint das Menue sofort. Bei DOS 2.0S müssen die Hilfsprogramme erst von der Diskette geladen werden. Deshalb müssen Sie bei Verwendung von DOS 2.0S einige Sekunden warten, bevor das Menue auf dem Bildschirm erscheint.
Es können jetzt keine BASIC-Befehle mehr, sondern nur noch die Diskettenhilfsprogramme ausgeführt werden. Wenn sich keine ROM-Kassette im Computer befindet und Sie das DOS booten, erscheint das DOS-Menue sofort auf dem Bildschirm (an Stelle von READY). Sobald Sie den DOS-Befehl geben, werden bei DOS 2.0S die Diskettenhilfsprogramme geladen. Sie sind in einem File namens DUP.SYS gespeichert. Eine Diskette mit diesem File muß in Laufwerk 1 eingelegt sein. Der Inhalt dieses Files wird in dem Speicherbereich abgelegt, in dem sich normalerweise Ihre BASIC-Programme befinden. Mit anderen Worten: das gerade im Speicher befindliche BASIC-Programm wird gelöscht. Wenn Sie vom

DOS-Menue aus wieder ins BASIC zurückgehen, werden Sie feststellen, daß der Speicher leer ist, gerade so, als hätten Sie einen NEW-Befehl ausgeführt. Falls Sie das Programm vor Ausführung des DOS-Befehls nicht auf Kassette oder Diskette gespeichert haben, ist es unwiderruflich verlorengegangen. Wir werden später allerdings noch einen Weg kennenlernen, wie BASIC-Programme bei Ausführung des DOS-Befehls geschützt werden können.
Bei DOS 1.0 wird DUP.SYS nicht benötigt, denn die Hilfsprogramme wurden beim Bootvorgang mitgeladen und befinden sich bereits im Speicher. Für sie ist ein eigener Speicherbereich reserviert. Wenn Sie unter DOS 1.0 den DOS-Befehl ausführen, werden Ihre BASIC-Programme deshalb in keiner Weise beeinflußt.

```
DISK OPERATING SYSTEM II  VERSION 2.0S
COPYRIGHT 1980 ATARI

A. DISK DIRECTORY I. FORMAT DISK
B. RUN CARTRIDGE  J. DUPLICATE DISK
C. COPY FILE      K. BINARY SAVE
D. DELETE FILE(S) L. BINARY LOAD
E. RENAME FILE    M. RUN AT ADDRESS
F. LOCK FILE      N. CREATE MEM.SAV
G. UNLOCK FILE    O. DUPLICATE FILE
H. WRITE DOS FILES
```

Bild 7-5: DOS-Menue (DOS 2.0S)

```
DISK OPERATING SYSTEM    9/24/79
COPYRIGHT 1979 ATARI

A. DISK DIRECTORY I. FORMAT DISK
B. RUN CARTRIDGE  J. DUPLICATE DISK
C. COPY FILE      K. BINARY SAVE
D. DELETE FILE(S) L. BINARY LOAD
E. RENAME FILE    M. RUN AT ADDRESS
F. LOCK FILE      N. DEFINE DEVICE
G. UNLOCK FILE    O. DUPLICATE FILE
H. WRITE DOS FILE
```

Anmerkung: Funktion N (DEFINE DEVICE) kann nicht ausgeführt werden.

Bild 7-6: DOS-Menue (DOS 1.0)

Bestimmung der DOS-Version

Wenn Sie den DOS-Befehl geben und das DOS-Menue sofort erscheint, arbeiten Sie mit DOS 1.0. Wenn jedoch vorher Laufwerk 1 anläuft, arbeiten Sie mit DOS 2.0S. Bei DOS 2.0S können Sie außerdem die Versionsnummer dem Menue direkt entnehmen. Sie ist rechts oben vermerkt (Bild 7-5).

Das MEM.SAVE-File

Wenn unter DOS 2.0S der DOS-Befehl gegeben wird, werden die Diskettenhilfsprogramme von der Diskette geladen. Dabei wird ein Teil des Speicherbereichs, in dem BASIC-Programme gespeichert werden, überschrieben. In diesem Abschnitt wird behandelt, wie der Inhalt dieses Speicherbereiches mit Hilfe des MEM.SAV-Files erhalten werden kann. Für DOS 1.0 ist das nicht relevant. Deshalb können Sie diesen Abschnitt überspringen, wenn Sie nur mit dieser DOS-Version arbeiten.
Unter DOS 2.0S wird bei Ausführung des DOS-Befehls zunächst geprüft, ob sich auf der in Laufwerk 1 eingelegten Diskette ein File namens MEM.SAV befindet. Wenn dieses File vorhanden ist, speichert der Computer alles, was sich in dem Speicherbereich befindet, der von den Diskettenhilfsprogrammen belegt werden wird, unter diesem Filenamen ab. Erst danach werden die Diskettenhilfsprogramme vom File DUP.SYS geladen und das DOS-Menue auf dem Bildschirm angezeigt. Wenn Sie zu einem späteren Zeitpunkt wieder das BASIC aufrufen, wird der Inhalt von MEM.SAV wieder in den Speicher zurückgeladen. Damit ist der Zustand vor Ausführung des DOS-Befehls wieder hergestellt. Ihre Programme befinden sich also nach wie vor im Speicher.
Bei Benutzung des MEM.SAV-Files müssen Sie allerdings einen Nachteil in Kauf nehmen: das Abspeichern und Laden des Files braucht seine Zeit. Normalerweise werden die Diskettenhilfsprogramme innerhalb von zehn Sekungen geladen. Falls ein MEM.SAVE-File vorhanden ist, werden noch einmal 20 Sekunden zum Abspeichern des Programmspeichers benötigt. Es dauert dann also drei mal so lange, bis das DOS-Menue auf dem Bildschirm erscheint. Wenn Sie später wieder das BASIC aufrufen, dauert es noch einmal sieben Sekunden, bis der Programmspeicher wieder geladen ist.
Wenn der DOS-Befehl gegeben wird, muß in Laufwerk 1 eine Diskette eingelegt sein, auf der sich sowohl DUP.SYS als auch MEM.SAV befindet. Sollten Sie eine Diskette nur für Ihre eigenen Files verwenden wollen, dann müssen Sie DUP.SYS und MEM.SAV. eventuell zusammen mit DOS.SYS, auf einer separaten Diskette speichern und diese Diskette vor Ausführung des DOS-Befehls in Laufwerk 1 einlegen. Auch wenn Sie das DOS-Menue wieder verlassen, muß diese Diskette in Laufwerk 1 eingelegt sein. Anderenfalls können die in MEM.SAV abgelegten Programme oder Programmteile nicht wieder geladen werden.
MEM.SAV ist kein gewöhnliches File. Um dieses File, mit dessen Hilfe der Inhalt des Speicherbereiches, in den die Diskettenhilfsprogramme geladen werden, gerettet und später wieder geladen wird, überhaupt anlegen zu können, gibt es ein eigenes Diskettenhilfsprogramm. Eine Bescheibung dieses Hilfsprogrammes folgt später.
Das MEM.SAV-File funktioniert auch, wenn sich kein BASIC-Programm in dem besagten Speicherbereich befindet. Der oben beschriebene Vorgang erfolgt immer, ganz gleich, was in diesem Speicherbereich gespeichert wurde.

Mehrdeutige Filenamen

Bei der Benutzung der Diskettenhilfsprogramme müssen sehr oft Filenamen eingegeben werden. Sie können dann einen ganz bestimmten oder auch einen mehrdeutigen Filenamen angeben. Letzteres kann manchmal sehr von Nutzen sein. Es kann zum Beispiel einmal vorkommen, daß Sie sich nur noch an die ersten drei Buchstaben eines Filenamens erinnern. Sie können den Filenamen dann so eingeben, daß das Disk Operating System

ein File heraussucht, dessen Filename mit diesen drei Buchstaben beginnt. Auch wenn Sie Files von einer Diskette auf eine andere kopieren wollen, kann dieser Vorgang durch die Angabe eines mehrdeutigen Filenamens sehr vereinfacht werden. Sie könnten zum Beispiel unter DOS 2.0S alle Files, deren Filenamen einen bestimmten Zusatz trägt, auf einmal kopieren, ohne jeden Filenamen einzeln eingeben zu müssen.
Der Stern (*) und das Fragezeichen (?) sind für das Disk Operating System sogenannte "wild card"-Zeichen. Der Stern kann für einen Filenamen, den Zusatz des Filenamens oder einen Teil davon stehen. Wenn Sie zum Beispiel als Filenamen *.BAS angeben, dann sind damit alle Files, deren Filename den Zusatz .BAS trägt, gemeint. Entsprechend steht DATAFILE.* für alle Files mit dem Filenamen DATAFILE. Der Zusatz des Filenamens spielt hierbei keine Rolle.
Innerhalb eines Filenamens sind alle Zeichen, die rechts von einem Stern stehen, irrelevant.
Zum Beispiel hat *FILE.TXT dieselbe Bedeutung wie *.TXT. Demgegenüber werden mit SPIEL*.BAS nur die Files angesprochen, deren Filenamen mit SPIEL beginnt und den Zusatz .BAS trägt. Beim Zusatz des Filenamens hat der Stern gerade die umgekehrte Wirkung: alle Zeichen, die links von einem Stern stehen, sind irrelevant. .A* hat also dieselbe Bedeutung wie .*, aber mit .*A werden nur die Files angesprochen, bei denen der Zusatz des Filenamens mit A endet.
Das Fragezeichen repräsentiert nur einen einzigen Buchstaben. SPIEL?.BAS steht zum Beispiel für alle Files, deren Filename aus sechs Buchstaben besteht, mit den fünf Buchstaben SPIEL beginnt und den Zusatz .BAS trägt.
Der Filename ????????.TXT ist gleichbedeutend mit *.TXT, denn für jeden der acht Buchstaben, aus denen ein Filenamen bestehen kann, ist ein Fragezeichen angegeben. Also werden alle Files, deren Filenamen den Zusatz .TXT trägt, dadurch angesprochen.
Wenn Sie bei der Benennung Ihrer Files systematisch vorgehen, dann werden sich die "wild card"-Zeichen bei der Programmierung als sehr nützlich erweisen, besonders dann, wenn Sie auch die in Tabelle 7-1 angegebenen Zusätze verwenden.

Das DOS-MENUE

Im DOS-Menue (Bild 7-5 und 7-6) sind viele Funktionen aufgeführt, mit deren Hilfe auf eine Diskette zugegriffen werden kann. Zunächst werden die Funktionen beschrieben, die für BASIC-Programmierer am interessantesten sind. Andere Funktionen werden nur dann benutzt, wenn in Maschinensprache programmiert wird. Sie werden am Ende des Kapitels beschrieben.
Zusammen mit dem Menue von DOS 2.0S erscheint die folgende Information auf dem Bildschirm:

```
SELECT ITEM OR  RETURN  FOR MENU
```

Unter DOS 1.0 sieht das ähnlich aus:

```
SELECT ITEM
```

Sie können nun eine der im DOS-Menue aufgeführten Funktionen wählen. Jede Funktion ist durch einen vorgestellten Buchstaben gekennzeichnet. Geben Sie diesen Buchstaben ein, um die entsprechende Funktion zu wählen und drücken Sie RETURN. Danach gehen Sie so vor, wie es in den nachfolgenden Abschnitten beschrieben ist. Bei den meisten Funktionen folgen nun noch weitere Eingaben wie zum Beispiel ein Filename oder eine Laufwerksnummer. Wenn sie keinen Buchstaben eingeben, sondern nur RETURN drücken, wird das DOS-Menue erneut angezeigt.
Alle Funktionen können mit jeder beliebigen Diskette, die Sie in Ihrem Computersystem

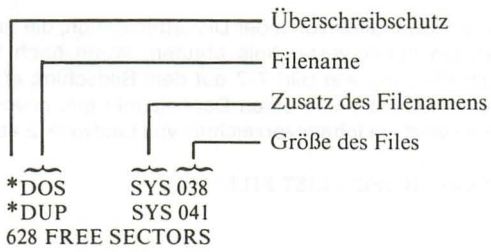
verwenden, ausgeführt werden. Sobald die "BUSY"-Kontrollampe Ihrer Diskettenstation verloschen ist, können Sie Disketten beliebig wechseln. Wenn Sie eine Funktion gewählt haben, können Sie diese Funktion jederzeit durch Drücken der BREAK-Taste unterbrechen. Danach werden Sie aufgefordert, erneut eine Funktion zu wählen.
Wenn Sie zu irgendeinem Zeitpunkt die SYSTEM RESET-Taste drücken, übernimmt das BASIC die Kontrolle über das System. Falls die BASIC-Kassette nicht eingesteckt ist, wird das DOS erneut gebootet und das DOS-Menue erscheint auf dem Bildschirm.

Der Inhalt einer Diskette

Wenn Sie die Namen aller Files, die sich auf einer Diskette befinden, abrufen wollen, wählen Sie Funktion A des DOS-Menues (DISK DIRECTORY = Inhaltsverzeichnis der Diskette). Nachdem Sie das getan haben, erscheint eine Meldung auf dem Bildschirm (Bild 7-7). Wenn Sie daraufhin RETURN drücken, wird der Computer die Namen aller Files, die auf der in Laufwerk 1 eingelegten Diskette gespeichert sind, auf dem Bildschirm anzeigen. Wenn Sie mit DOS 2.0S arbeiten, sind auf jeden Fall die beiden Systemfiles DOS.SYS und DUP.SYS auf der Diskette gespeichert (Bild 7-8). Bei DOS 1.0 fehlt DUP.SYS.

```
DIRECTORY--SEARCH SPEC, LIST FILE?
```

Bild 7-7: Meldung bei Wahl von Funktion A (DISK DIRECTORY)



Anmerkung: Es können noch weitere Files aufgeführt sein. Bei DOS 1.0 gibt es das File DUP.SYS nicht. Auch in einigen anderen Punkten unterscheiden sich die Inhaltsverzeichnisse voneinander.

Bild 7-8: Inhaltsverzeichnis der Masterdiskette (DOS 2.0S)

Die einzelnen Teile des Inhaltsverzeichnisses

In jeder Zeile des Inhaltsverzeichnisses werden Angaben über ein File gemacht. Diese Angaben betreffen den Überschreibschutz, den Filenamen, den Zusatz des Filenamens und die Filegröße (Bild 7-8).
Ob ein File gegen Überschreiben geschützt ist, wird durch einen Stern vor dem Filenamen angezeigt. Wenn dieser Stern vorhanden ist, ist das entsprechende File schreibgeschützt, das heißt, es kann nicht verändert oder gelöscht werden. Damit lassen sich besonders wertvolle Files gegen versehentliches Überschreiben oder Löschen schützen. Dieser Überschreibschutz kann mit Hilfe von anderen Funktionen des DOS-Menues gesetzt und wieder gelöscht werden. Diese Funktionen werden in Kürze beschrieben.
Der Filename und sein Zusatz (falls vorhanden) werden im Inhaltsverzeichnis direkt nebeneinander geschrieben. Die sonst übliche Trennung durch einen Punkt erfolgt hier nicht. Wenn Sie allerdings einen Filenamen mit einem Zusatz eingeben, muß dieser Punkt vorhanden sein. Sowohl die Filenamen als auch die Zusätze werden rechtsbündig untereinandergeschrieben.
Die Filelänge wird rechts neben dem Filenamen angegeben. Die Zahl, die dort steht, gibt die Anzahl der Blöcke zu jeweils 128 Zeichen an, die das jeweilige File auf der Diskette belegt. Je größer das File ist, desto mehr Blöcke werden belegt. Ein Block wird für die Speicherung eines Files mindestens benötigt.
In der letzten Zeile eines Inhaltsverzeichnisses wird angegeben, für wieviel Blöcke auf der Diskette noch Platz vorhanden ist. Unter DOS 2.0S werden das bei einer leeren Diskette 707 und unter DOS 1.0 709 sein. Diese Zahl wird folgendermaßen berechnet: zunächst werden alle Filelängen zusammengezählt und diese Zahl dann von 707 (DOS 2.0S) beziehungsweise 709 (DOS 1.0) abgezogen. Wenn Sie ein File speichern wollen, das mehr Platz belegt als auf der Diskette noch vorhanden ist, erfolgt eine Fehlermeldung ('DISK FULL').

Abruf des Inhaltsverzeichnisses von einem beliebigen Laufwerk

Sie können selbstverständlich von jeder Diskettenstation, die an den ATARI-Computer angeschlossen ist, ein Inhaltsverzeichnis abrufen. Wenn nach Wahl von Funktion A des DOS-Menues die Meldung von Bild 7-7 auf dem Bildschirm erscheint, geben Sie die gewünschte Laufwerksnummer und einen Doppelpunkt ein, bevor Sie RETURN drücken. Im folgenden Beispiel wird ein Inhaltsverzeichnis von Laufwerk 2 abgerufen:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
D2:
```

Der Buchstabe D vor der Laufwerksnummer muß nicht unbedingt eingegeben werden.

Abruf von einem Teil des Inhaltsverzeichnisses

Es ist durchaus möglich, nur einen Teil des Inhaltsverzeichnisses abzurufen. Es werden dann nur die Filenamen angezeigt, die bestimmten, von Ihnen vorgegebenen Bedingungen genügen. Diese Bedingungen werden mit Hilfe eines mehrdeutigen Filenamens aufgestellt. Es könnten zum Beispiel nur die Filenamen angezeigt werden, die mit dem Buchstaben E beginnen:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
E*.*
```

Wenn das Disk Operating System die in Bild 7-7 dargestellte Meldung ausgibt, müßten Sie

im Prinzip zwei Dinge eingeben: erstens, welches Laufwerk auf welche Filenamen hin untersucht werden soll (SEARCH SPEC(IFICATION)) und zweitens, wo die gefundenen Filenamen ausgegeben werden sollen (LIST FILE). Wenn Sie nur RETURN drücken, werden alle Files von Laufwerk 1 auf dem Bildschirm angezeigt. Dies sind die Voreinstellungen, die im Computer gespeichert sind.
Falls Sie einen mehrdeutigen Filenamen für SEARCH SPEC eingeben, dann ersetzt diese Eingabe die Voreinstellung. Wenn zum Beispiel nur die Filenamen mit dem Zusatz .BAS angezeigt werden sollen, dann geben Sie folgendes ein:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
D1:*.BAS
```

D1: entspricht einer Voreinstellung, also können Sie es auch weglassen:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
*.BAS
```

In beiden Fällen wird Laufwerk 1 auf Filenamen untersucht, die den Zusatz .BAS tragen. Diese Filenamen werden dann auf dem Bildschirm angezeigt.
Es ist sogar möglich, einen ganz bestimmten Filenamen anzugeben, nach dem Sie suchen. Wenn sich das File nicht auf der Diskette befindet wird im Inhaltsverzeichnis kein Filename angegeben. Es wird dann nur angezeigt, wieviel Platz noch auf der Diskette vorhanden ist.

Ausgabe des Inhaltsverzeichnisses auf einem beliebigen Peripheriegerät

Die in Bild 7-7 aufgeführte Meldung fordert Sie auch dazu auf, einzugeben, auf welchem Peripheriegerät die Ausgabe der gefundenen Filenamen erfolgen soll. Bisher haben wir das nicht getan. Das Disk Operating System brachte das Inhaltsverzeichnis entsprechend der Voreinstellung immer auf den Bildschirm. Falls die Ausgabe woanders erfolgen soll, geben Sie ein Komma, den Namen des gewünschten Peripheriegerätes und einen Doppelpunkt ein und drücken RETURN. Wenn das Inhaltsverzeichnis von Laufwerk 1 auf dem Drucker ausgegeben werden soll, sieht das folgendermaßen aus:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
D1:*,*,P:
```

Die Eingabe für SEARCH SPEC können Sie genausogut weglassen, denn sie entspricht den Voreinstellungen. Gleichbedeutend mit der obigen Eingabe ist die folgende:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
,P:
```

Vergessen Sie nicht das Komma vor dem P:. Damit wird dem Disk Operating System mitgeteilt, daß es sich bei dem folgenden um die zweite gewünschte Eingabe (LIST FILE) handelt. Wenn Sie das Komma vergessen, nimmt das Disk Operating System an, daß alle Files angezeigt werden sollen, die auf der im angegebenen Gerät eingelegten Diskette gespeichert sind. Das angegebene Gerät ist in diesem Fall der Drucker; die gestellte Aufgabe kann also unmöglich gelöst werden.
Wenn Sie das Inhaltsverzeichnis von einem anderen Laufwerk ausdrucken lassen wollen, geben Sie vor dem Komma die Laufwerksnummer und einen Doppelpunkt an:

```
DIRECTORY--SEARCH SPEC, LIST FILE?
2:,P:
```

Verlassen des DOS-Menues

Wenn Sie das DOS-Menue verlassen wollen und die eingesteckte ROM-Kassette wieder die Kontrolle über den Computer übernehmen soll, wählen Sie Funktion B des DOS-Menues (RUN CARTRIDGE). Es ist auch möglich, die SYSTEM RESET-Taste zu drücken. Wenn die BASIC-Kassette eingesteckt ist, erscheint anschließend READY und bei der Editor/Assembler-Kassette EDIT auf dem Bildschirm. Falls keine ROM-Kassette eingesteckt ist, erscheint die Meldung 'NO CARTRIDGE'. Anschließend können sie eine andere Funktion des DOS-Menues wählen.
Wenn Sie mit einem MEM.SAV-File arbeiten (nur bei DOS 2.0S), benutzen Sie nicht Funktion B des DOS-Menues, sondern drücken Sie immer die SYSTEM RESET-Taste. So wird sichergestellt, daß der mit Hilfe des MEM.SAV-Files gerettete Speicherinhalt auch wieder fehlerfrei geladen wird.

Kopieren von Files

Wenn Sie den Inhalt eines Files in ein anderes File kopieren wollen, wählen Sie Funktion C des DOS-Menues (COPY FILE). Solange die Filenamen nicht gleich sind, dürfen beide Files auf derselben Diskette sein. Wenn das nicht der Fall ist, kann nur dann kopiert werden, wenn auf beide Disketten gleichzeitig zugegriffen werden kann. Mit anderen Worten: zum Kopieren eines Files von einer Diskette auf eine andere brauchen Sie zwei Diskettenstationen. Falls Sie nur eine Diskettenstation haben und von einer Diskette auf eine andere kopieren wollen, müssen Sie das mit Hilfe von Funktion O des DOS-Menues tun (DUPLICATE FILE).
Nachdem Sie Funktion C des DOS-Menues gewählt haben, erscheint eine Meldung auf dem Bildschirm. Sie werden aufgefordert, das Laufwerk und den Filenamen für Quelle und Ziel einzugeben, beispielsweise so:

```
COPY FILE--FROM,TO?
FILE1.BAS,FILE1.BAK
```

Da bei keinem der beiden Files eine Laufwerksnummer angegeben wurde, setzt das Disk Operating System für beide Laufwerk 1 ein. Beachten Sie das Komma zur Trennung der Filenamen.
Wenn das Zielfile bereits existiert, wird es mit dem Inhalt des Quellfiles überschrieben. Anderenfalls wird ein File mit dem angegebenen Namen angelegt. Wenn das Quellfile nicht existiert, tritt ein Fehler auf (ERROR 170). Außerdem treten Fehler auf, wenn auf der Zieldiskette nicht mehr genug Platz vorhanden ist (ERROR 162) oder wenn das Inhaltsverzeichnis der Zieldiskette voll ist (ERROR 169).
Eine weitere Meldung wird ausgegeben, wenn sie mit dem MEM.SAV-File arbeiten (nur bei DOS 2.0S). Sie müssen die Frage beantworten, ob der gesamte Speicherbereich, in dem normalerweise Programme gespeichert werden, zum Kopieren benutzt werden darf (Bild 7-9). Wenn Sie diese Frage durch Eingabe von Y und Drücken der RETURN-Taste bejahen, dann wird der Inhalt des MEM.SAV-Files nach Verlassen des DOS-Menues nicht wieder geladen und der gerettete Speicherbereich nicht wiederhergestellt. Bei jeder anderen Eingabe wird das Kopierprogramm abgebrochen und die Funktion des MEM.SAV-Files bleibt voll erhalten. Es gibt allerdings einen Weg, wie Sie trotzdem mit dem Kopierprogramm arbeiten können. Sie verlassen einfach das DOS-Menue und retten das im Speicher befindliche Programm auf Kassette oder Diskette. Danach ist es unerheblich, ob das MEM.SAV-File außer Funktion gesetzt wird, denn Sie können das gerettete Programm jederzeit wieder laden.
DOS.SYS hat ein besonderes Format und kann mit Hilfe des Kopierprogrammes (Funktion C des DOS-Menues) nicht kopiert werden. Wenn Sie es versuchen, wird ein Fehler auftreten.

```
TYPE "Y" IF OK TO USE PROGRAM AREA
CAUTION: A "Y" INVALIDATES MEM.SAV
```

Bild 7-9: Nachfrage zur Benutzung des Programmspeichers

Kopieren von mehreren Files auf einmal

Unter DOS 2.0S können Sie für das Quellfile einen mehrdeutigen Filenamen verwenden. In diesem Fall dürfen Sie, wie das folgende Beispiel zeigt, für das Zielfile keinen Filenamen, sondern nur eine Laufwerksnummer angeben:

```
COPY FILE--FROM,TO?
*.BAK,D2:
```

Hiermit werden alle Files, deren Filenamen den Zusatz .BAK trägt, von Laufwerk 1 auf Laufwerk 2 kopiert.
Files, deren Filename den Zusatz .SYS trägt, werden bei Angabe eines mehrdeutigen Filenamens nicht kopiert. Wenn Sie ein solches File kopieren wollen, müssen Sie den Filenamen vollständig ohne Verwendung von "wild card"-Zeichen eingeben.

Zusammenfügen von zwei Files mit Hilfe des Kopierprogrammes

Unter DOS 2.0S können mit Hilfe des Kopierprogrammes (Funktion C des DOS-Menues) zwei Files zusammengefügt werden. Wenn Sie das tun wollen, schreiben Sie hinter den Filenamen des Zielfiles zusätzlich diese zwei Zeichen: /A (A = Append = Anfügen). Die folgende Eingabe ist ein Beispiel dafür:

```
COPY FILE--FROM,TO?
D2:NAME.TXT,D3:ZAHL.TXT/A
```

Das Zielfile muß bereits existieren. Allerdings wird es jetzt nicht mit dem Inhalt des Quellfiles überschrieben, sondern der Inhalt des Quellfiles wird am Ende des Zielfiles angefügt.
Es ist jedoch nicht möglich, zwei BASIC-Programme, die mit Hilfe des SAVE-Befehls gespeichert wurden, zusammenzufügen. Auf der Diskette wird das zwar funktionieren, aber wenn Sie später einen LOAD-Befehl geben, um die beiden Programme zu laden, erkennt der Computer nicht, daß an das erste Programm noch ein zweites Programm angefügt wurde.
Sie können aber problemlos BASIC-Programme zusammenfügen, die unter Verwendung des LIST-Befehls gespeichert wurden. Das Quellfile wird an das Zielfile angefügt und wenn Sie später das Zielfile mit Hilfe des ENTER-Befehls laden, werden beide Programme in den Speicher übernommen. Das Ergebnis sieht so aus, als wenn sich das im Zielfile gespeicherte Programm bereits im Hauptspeicher des Computers befunden und Sie dann jede Zeile des im Quellfile gespeicherten Programmes zusätzlich eingegeben hätten. Das heißt also, daß eine Zeile des ersten Programmes (Zielfile), die dieselbe Zeilennummer wie eine Zeile des zweiten Programmes (Quellfile) trägt, von dieser überschrieben wird. Am

besten speichern Sie das Ergebnis dieses Ladevorganges dann noch einmal mit Hilfe des LIST-Befehls unter dem Filenamen des Zielfiles ab, damit auf der Diskette kein Platz durch zwei Zeilen mit der gleichen Zeilennummer verschwendet wird.

Kopieren mit einer Diskettenstation und zwei Disketten

Wenn Sie ein File von einer Diskette auf eine andere kopieren wollen und nur eine Diskettenstation an Ihren ATARI-Computer angeschlossen ist, dann wählen Sie Funktion O des DOS-Menues (DUPLICATE FILE). Danach müssen sie den Namen des Files eingeben, das sie kopieren wollen:

```
NAME OF FILE TO MOVE?
```

Sie brauchen nur einen Filenamen anzugeben, weil das Zielfile bei diesem Kopierprogramm grundsätzlich den gleichen Namen wie das Quellfile bekommt. Auch eine Laufwerksnummer brauchen Sie nicht einzugeben. Sowohl als Quell- als auch als Ziellaufwerk wird Laufwerk 1 verwendet.
Es ist durchaus möglich, im Filenamen 'wild card'-Zeichen zu verwenden. In diesem Fall wird ein File nach dem anderen kopiert, das diesem Filenamen entspricht. Allerdings können Files, deren Filenamen den Zusatz .SYS trägt, so nicht kopiert werden. Wenn Sie ein solches File kopieren wollen, müssen Sie den Filenamen vollständig ohne Verwendung von 'wild card'-Zeichen eingeben.
Falls Sie mit einem MEM.SAV-File arbeiten (nur unter DOS 2.0S), erscheint als nächstes die Frage, ob der gesamte Programmspeicherbereich zum Kopieren benutzt werden darf (Bild 7-9). Wenn sie diese Frage durch Eingabe von Y und Drücken der RETURN-Taste bejahen, wird der Inhalt des MEM.SAV-Files nach Verlassen des DOS-Menues nicht wieder geladen und das im Programmspeicher befindliche Programm nicht wiederhergestellt. Bei jeder anderen Eingabe bleibt die Funktion des MEM.SAV-Files vollständig erhalten. Der Kopiervorgang findet zwar trotzdem statt, aber es wird sehr viel mehr Zeit dafür benötigt.
Nachdem Sie diese Frage beantwortet haben, kann der Kopiervorgang beginnen. Auf dem Bildschirm wird angezeigt, wann Sie die Quelldiskette (Source Disk) und wann Sie die Zieldiskette (Destination Disk) einlegen müssen. Es kann vorkommen, daß ein mehrmaliger Wechsel der Disketten notwendig ist. Jedesmal, nachdem Sie eine Diskette eingelegt haben, muß die RETURN-Taste gedrückt werden.
Nach Einlegen der Quelldiskette liest der Computer das Quellfile oder einen Teil davon, falls das File sehr umfangreich ist. Nach Einlegen der Zieldiskette wird das Zielfile beziehungsweise ein Teil davon abgespeichert. Wenn das File sehr groß ist oder wenn Sie mehr als ein File kopieren wollen, müssen Sie die Disketten mehrmals wechseln.
Vorsicht: Falls Sie unter DOS 1.0 Funktion O des DOS-Menues wählen, wird der Programmspeicher grundsätzlich zum Kopieren benutzt. Wenn Sie nach einem Kopiervorgang wieder das BASIC aufrufen, wird jedes BASIC-Programm gelöscht sein.

Löschen von Files

Nachdem Sie eine gewisse Zeit mit einer Diskette gearbeitet haben, werden sich auf dieser Diskette mit Sicherheit einige Files befinden, die Sie nicht mehr benötigen. Wenn Sie diese Files löschen, haben Sie wieder mehr Platz zur Speicherung neuer Files. Wählen Sie dafür Funktion D des DOS-Menues (DELETE FILE(S)). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
DELETE FILE SPEC
```

Sie müssen nun die Laufwerksnummer und den Filenamen des Files eingeben, das Sie lö-

schen wollen. Dabei können Sie auch 'wild card'-Zeichen verwenden, wenn mehrere Files gelöscht werden sollen. Nachdem Sie das getan haben, erscheint eine neue Meldung auf dem Bildschirm:

```
TYPE "Y" TO DELETE...
```

Danach wird auf der Diskette nach Filenamen gesucht, die Ihrer Eingabe entsprechen. Wenn ein solcher Filenamen gefunden wurde, wird er, gefolgt von einem Fragezeichen, auf dem Bildschirm angezeigt. Falls Sie das File wirklich löschen wollen, geben Sie zur Bestätigung Y ein und drücken die RETURN-Taste. Bei jeder anderen Eingabe wird das File nicht gelöscht.
Ein schreibgeschütztes File kann nicht gelöscht werden. Wenn Sie es versuchen, tritt ein Fehler auf (ERROR 167). Danach können Sie eine andere Funktion des DOS-Menues wählen.
Wenn Sie eine Diskette ganz löschen wollen, geben Sie folgendes ein:

```
DELETE FILE SPEC
*.*
```

Löschen von Files ohne Bestätigung der zu löschenden Filenamen

Normalerweise muß bei Funktion D des DOS-Menues jeder Filename vor dem eigentlichen Löschvorgang noch einmal bestätigt werden. Damit wird verhindert, daß Sie aus Versehen einmal das falsche File löschen. Allerdings ist es einigermaßen lästig, jedesmal Y einzugeben und die RETURN-Taste drücken zu müssen, wenn Sie unter Verwendung eines mehrdeutigen Filenamens eine größere Anzahl von Files auf einmal löschen wollen. In solchen Fällen wäre es zweckmäßig, wenn nicht jeder einzelne Filename bestätigt werden muß. Sie erreichen das, indem Sie bei der Eingabe des zu löschenden Filenamens die folgenden zwei Zeichen anfügen: /N. Im folgenden Beispiel werden alle Files von der in Laufwerk 1 eingelegten Diskette gelöscht, ohne daß jeder Filename noch einmal angezeigt wird.

```
DELETE FILE SPEC
*.*/N
```

Seien Sie bei der Verwendung dieses Zusatzes besonders vorsichtig, denn ein einmal gelöschtes File ist für immer verloren.

Ändern von Filenamen

Wenn Sie den Namen eines Files ändern wollen, wählen Sie Funktion E des DOS-Menues (RENAME FILE). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
RENAME - GIVE OLD NAME, NEW
```

Geben Sie zuerst den alten Filenamen und dann, nach einem Komma, den neuen Filenamen ein. Weil das File lediglich einen neuen Namen bekommt, können Sie nur für den alten Filenamen eine Laufwerksnummer eingeben. Zum Kopieren auf ein anderes Laufwerk müssen Sie Funktion C des DOS-Menues verwenden.
Falls ein File mit dem angegebenen alten Filenamen nicht existiert, tritt ein Fehler auf (ERROR 170). Falls das File schreibgeschützt ist, tritt ein anderer Fehler auf (ERROR 167).
Vorsicht: Wenn ein File mit dem angegebenen neuen Filenamen bereits existiert, haben Sie anschließend zwei Files mit demselben Namen auf einer Diskette. Ein Zugriff auf nur

eines der beiden Files ist dann nicht mehr möglich, es können nur noch beide Files gleichzeitig angesprochen werden. Bei dem Versuch, den Filenamen von einem File zu ändern, wird auch der Filename des zweiten geändert. Der einzige Ausweg aus diesem Dilemma besteht darin, beide Files zu löschen. Wenn Sie sich nicht ganz sicher sind, welche Filenamen Sie schon benutzt haben, sehen Sie sich deshalb mit Hilfe von Funktion A des DOS-Menues das Inhaltsverzeichnis einer Diskette an, bevor Sie den Namen eines Files ändern. Wie das folgende Beispiel zeigt, können auch mehrdeutige Filenamen angegeben werden:

```
RENAME - GIVE OLD NAME, NEW
*.DAT,*.TXT
```

Hiermit werden nur die Filenamen verändert, deren Zusatz .DAT lautet. Dieser Zusatz wird geändert und lautet nach Ausführung der Änderung .TXT. Im folgenden Beispiel wird der Zusatz aller Filenamen geändert in .ZZZ, ganz gleich, wie der Zusatz bisher lautete:

```
RENAME - GIVE OLD NAME, NEW
*.*,*.ZZZ
```

Seien Sie sehr vorsichtig, wenn Sie beim Ändern von Filenamen 'wild card'-Zeichen benutzen. Nur allzu schnell haben Sie zwei Files mit demselben Namen auf einer Diskette.
Ändern Sie niemals den Namen von DOS.SYS. Anderenfalls kann diese Diskette nicht mehr gebootet werden. Wenn Sie den Namen von DUP.SYS ändern (nur bei DOS 2.0S), kann das DOS-Menue von dieser Diskette nicht mehr geladen werden (bei Ausführung des DOS-Befehls).

Files gegen Überschreiben schützen

Der Inhalt eines schreibgeschützten Files einschließlich des Filenamens kann nicht geändert oder gelöscht werden. Wenn Sie ein File gegen Überschreiben schützen wollen, wählen Sie Funktion F des DOS-Menues (LOCK FILE). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
WHAT FILE TO LOCK?
```

Geben Sie nun die Laufwerksnummer und den Filenamen des Files an, das schreibgeschützt werden soll. Im folgenden Beispiel ist das das File mit dem Namen DOS.SYS auf der in Laufwerk 1 eingelegten Diskette:

```
WHAT FILE TO LOCK?
DOS.SYS
```

Auch hier können Sie bei der Eingabe des Filenamens 'wild card'-Zeichen verwenden, falls Sie mehrere Files auf einmal schützen wollen. Wenn Sie als Filenamen zum Beispiel *.* eingeben, werden alle Files geschützt, bei Eingabe von *.BAS nur die Files, deren Filenamen den Zusatz .BAS tragen.
Häufig wird DOS.SYS und, falls vorhanden, DUP.SYS gegen Überschreiben geschützt. Damit wird verhindert, daß eines dieser beiden Systemfiles versehentlich verändert oder gelöscht wird.

Den Schreibschutz eines Files entfernen

Wenn Sie ein schreibgeschütztes File wieder beschreibbar machen wollen, wählen Sie Funktion G des DOS-Menues (UNLOCK FILE). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
WHAT FILE TO UNLOCK?
```

Geben Sie nun die Laufwerksnummer und den Namen des Files ein, das wieder beschreibbar gemacht werden soll. Im folgenden Beispiel wird der Schreibschutz vom File mit dem Namen ADDRESS.DAT auf der in Laufwerk 2 eingelegten Diskette entfernt:

```
WHAT FILE TO UNLOCK?
D2:ADDRESS.DAT
```

Auch hier können Sie 'wild card'-Zeichen verwenden, wenn Sie mehrere Files auf einmal wieder beschreibbar machen wollen. Bei Angabe von *.* wird beispielsweise jedes File beschreibbar gemacht und bei Angabe von *.BAS nur die Files, deren Filenamen den Zusatz .BAS trägt.

Schreiben von neuen DOS-Files

Das Disk Operating System ist bei DOS 1.0 im File DOS.SYS gespeichert. Bei DOS 2.0S werden zur Speicherung zwei Files benötigt: DOS.SYS und DUP.SYS. Wenn Sie diese Files auf eine in einem bestimmten Laufwerk eingelegte Diskette kopieren wollen, wählen Sie Funktion H des DOS-Menues (WRITE DOS FILES). Die Files werden nicht von einer anderen Diskette, sondern direkt aus dem Hauptspeicher des Computers kopiert. Dieser Vorgang ist bei DOS 1.0 und DOS 2.0S nicht vollkommen gleich.

Schreiben der DOS-Files unter DOS 2.0S

Nachdem Sie Funktion H des DOS-Menues gewählt haben, erscheint die folgende Meldung auf dem Bildschirm:

```
DRIVE TO WRITE DOS FILES TO?
```

Geben Sie nun die Laufwerksnummer (1, 2, 3 oder 4) des Laufwerks ein, in dem sich die Diskette befindet, auf die die DOS-Files geschrieben werden sollen.
Diese Wahl müssen sie durch die Eingabe von Y noch einmal bestätigen:

```
DRIVE TO WRITE DOS FILES TO?
1
TYPE "Y" TO WRITE DOS TO DRIVE 1
```

Bei jeder anderen Eingabe wird Funktion H des DOS-Menues abgebrochen und Sie können eine andere Funktion wählen.

Schreiben der DOS-Files unter DOS 1.0

Unter DOS 1.0 werden die DOS-Files immer auf die in Laufwerk 1 eingelegte Diskette geschrieben. Hier ist es also nicht möglich, das Laufwerk zu wählen. Nach Wahl von Funktion

H des DOS-Menues erscheint die folgende Meldung auf dem Bildschirm:

```
TYPE "Y" TO WRITE NEW DOS FILE
```

Formatierung von Disketten

Bevor Sie eine neue Diskette benutzen können, müssen Sie sie formatieren. Dabei werden Markierungen auf die Diskette geschrieben, mit deren Hilfe das Disk Operating System feststellen kann, an welcher Stelle der Diskette der Schreib-Lese-Kopf sich gerade befindet. Bei der Formatierung einer Diskette werden alle Daten, die sich eventuell darauf befinden, gelöscht. Achten Sie also darauf, daß Sie niemals die falsche Diskette formatieren.
Zur Formatierung einer Diskette wählen Sie Funktion I des DOS-Menues (FORMAT DISK). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
WHICH DRIVE TO FORMAT?
```

Geben Sie nun die Nummer des Laufwerks ein, in dem sich die Diskette befindet, die formatiert werden soll. Da bei der Formatierung einer Diskette alle darauf befindlichen Daten gelöscht werden, müssen sie Ihre Wahl zur Sicherheit noch einmal bestätigen:

```
WHICH DRIVE TO FORMAT?
1
TYPE "Y" TO FORMAT DRIVE 1
```

Falls Sie versehentlich die falsche Diskette eingelegt haben, können Sie Ihren Irrtum jetzt immer noch korrigieren. Am besten überprüfen Sie noch einmal, ob Sie die richtige Diskette eingelegt haben.
Wenn Sie dann Y eingeben, wird die entsprechende Diskette formatiert. Dieser Vorgang dauert etwa eine Minute. Bei jeder anderen Eingabe wird Funktion I des DOS-Menues abgebrochen und Sie können eine andere Funktion wählen.

Kopieren von Disketten

Obwohl Sie mit Hilfe von Funktion C des DOS-Menues jedes gewünschte File kopieren können, wird es öfters einmal vorkommen, daß Sie eine Diskette ganz kopieren wollen. Zu diesem Zweck wählen Sie Funktion J des DOS-Menues (DUPLICATE DISK). Damit werden alle Files, die sich auf einer Diskette befinden, kopiert, auch die Files, deren Filenamen den Zusatz .SYS trägt. Es kann sowohl von einem Laufwerk auf ein anderes als auch mit nur einem Laufwerk kopiert werden.
Nach der Wahl dieser Funktion erscheint die folgende Meldung auf dem Bildschirm:

```
DUP DISK--SOURCE,DEST DRIVES?
```

Vorsicht: Unter DOS 1.0 wird zum Kopieren von Disketten grundsätzlich der Programmspeicher benötigt. Deshalb wird nach dem Kopiervorgang ein BASIC-Programm, das sich eventuell im Speicher befunden hat, gelöscht sein.

Kopieren von Disketten mit nur einem Laufwerk

Wenn nur eine Diskettenstation an Ihren ATARI-Computer angeschlossen ist, geben Sie folgendes ein:

```
DUP DISK--SOURCE,DEST DRIVES?
1,1
```

Danach erscheint eine andere Meldung:

```
INSERT SOURCE DISK, TYPE RETURN
```

Wenn Sie die Diskette, die Sie kopieren wollen, nicht bereits eingelegt haben, dann tun Sie das jetzt. Danach drücken Sie die RETURN-Taste. Der Computer liest nun einen Teil des Disketteninhalts in den Hauptspeicher ein. Als nächstes werden Sie aufgefordert, die Zieldiskette einzulegen:

```
INSERT DESTINATION DISK,TYPE RETURN
```

Legen Sie jetzt die Zieldiskette ein und drücken Sie die RETURN-Taste. Ein paar Sekunden später werden Sie wieder aufgefordert, die Quelldiskette (Source Disk) einzulegen. Sie müssen Quell- und Zieldiskette einige Male wechseln. Wie oft, das hängt davon ab, wieweit der Hauptspeicher Ihres Computers ausgebaut ist und wieviel Daten auf der Quelldiskette gespeichert sind. Dieses Wechselspiel wiederholt sich so oft, bis alle Daten der Quelldiskette auf die Zieldiskette kopiert worden sind.
Wenn Sie mit dem MEM.SAV-File arbeiten (nur unter DOS 2.0S), wird eine weitere Meldung auf dem Bildschirm erscheinen, nachdem Sie die Quelldiskette zum ersten Mal eingelegt haben. Sie müssen die Frage beantworten, ob der gesamte Speicherbereich, in dem normalerweise Programme gespeichert werden, zum Kopieren benutzt werden darf (Bild 7-9). Wenn Sie diese Frage durch Eingabe von Y und Drücken der RETURN-Taste bejahen, dann wird der Inhalt des MEM.SAV-Files nach Verlassen des DOS-Menues nicht wieder geladen und der gerettete Speicherbereich nicht wiederhergestellt. Bei jeder anderen Eingabe wird das Kopierprogramm abgebrochen und die Funktion des MEM.SAV-Files bleibt voll erhalten. Es gibt allerdings einen Weg, wie Sie trotzdem eine Diskette kopieren können. Sie verlassen einfach das DOS-Menue und retten das im Speicher befindliche Programm auf Kassette oder Diskette. Danach ist es unerheblich, ob das MEM.SAV-File außer Funktion gesetzt wird, denn sie können das gerettete Programm jederzeit wieder laden.

Kopieren von Disketten mit mehreren Laufwerken

Wenn mehrere Diskettenstationen an Ihren ATARI-Computer angeschlossen sind, ist es sehr viel einfacher, Disketten zu kopieren. Quell- und Zieldiskette werden in zwei verschiedene Laufwerke eingelegt. Während des Kopiervorganges brauchen Sie dann keine Diskette mehr zu wechseln. Im folgenden Beispiel wird der Inhalt der in Laufwerk 1 eingelegten Diskette auf die in Laufwerk 2 befindliche Diskette kopiert:

```
DUP DISK--SOURCE,DEST DRIVES?
1,2
INSERT BOTH DISKETTES, TYPE RETURN
```

Nachdem Sie die Laufwerksnummern angegeben haben, können Sie die Disketten einlegen, falls Sie es noch nicht getan haben. Dann drücken Sie die RETURN-Taste und der Kopiervorgang beginnt.

Auch hier erscheint eine weitere Meldung (Bild 7-9) auf dem Bildschirm, falls sie mit dem MEM.SAV-File arbeiten (nur unter DOS 2.0S). Wenn sie daraufhin Y eingeben und die RETURN-Taste drücken, wird der gesamte Programmspeicherbereich zum Kopieren benutzt. Bei jeder anderen Eingabe wird Funktion J des DOS-Menues abgebrochen und Sie können eine andere Funktion wählen.

Erstellen eines MEM.SAV-Files

Wenn Sie ein MEM.SAV-File erstellen wollen (nur unter DOS 2.0S), wählen Sie Funktion N des DOS-Menues (CREATE MEM.SAV). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
TYPE "Y" TO CREATE MEM.SAV
```

Das MEM.SAV-File wird grundsätzlich auf die in Laufwerk 1 eingelegte Diskette geschrieben. Nach Eingabe von Y wird das File erstellt. Bei jeder anderen Eingabe wird Funktion N des DOS-Menues abgebrochen.
Falls sich auf der in Laufwerk 1 eingelegten Diskette bereits ein MEM.SAV-File befindet, wird die folgende Meldung auf dem Bildschirm ausgegeben: MEM.SAV-FILE ALREADY EXISTS (= MEM.SAV-File bereits vorhanden) und die Funktion ebenfalls abgebrochen.

Abspeichern und Laden von BASIC-Programmen mit Hilfe der Diskettenstation

Es gibt fünf BASIC-Befehle, mit deren Hilfe Sie auf das Disk Operating System zugreifen können. SAVE und LIST verwenden Sie zum Speichern von Programmen und LOAD, ENTER und RUN zum Laden.

Abspeichern eines Programmes

Ein einmal eingegebenes Programm bleibt im Computer gespeichert bis er ausgeschaltet oder der Speicher gelöscht wird, zum Beispiel durch die Eingabe von NEW. Mit Hilfe des SAVE- und des LIST-Befehls können Sie Programme auf Diskette unter einem von Ihnen gewählten Filenamen abspeichern. Außer dem Filenamen und dem Zusatz des Filenamens müssen Sie die Nummer des Laufwerks angeben, in dem sich die Diskette befindet, auf der das Programm gespeichert werden soll. Bei Ausführung von einem der folgenden Befehle wird ein Programm auf Laufwerk 1 abgespeichert:

```
SAVE "D1:MYPROG.BAS"
LIST "D1:MYPROG.LST"
```

Der Gerätename (D: = Diskettenstation) muß immer angegeben werden; die Laufwerksnummer hingegen kann weggelassen werden. In diesem Fall wird grundsätzlich Laufwerk 1 angesprochen. Auch bei Ausführung der folgenden Befehle wird ein Programm auf Laufwerk 1 gespeichert:

```
SAVE "D:MENUE.BAS"
LIST "D:MENUE.LST"
```

Beachten Sie, daß bei der Speicherung mit Hilfe des SAVE-Befehls als Zusatz des Filenamens .BAS und bei Verwendung des LIST-Befehls .LST angegeben wurde. Dann kann zu einem späteren Zeitpunkt auf den ersten Blick erkannt werden, ob ein File mit LIST oder

SAVE gespeichert wurde. Das ist wichtig, weil das Aufzeichnungsformat nicht dasselbe ist.
Bei Verwendung des LIST-Befehls wird der ATASCII-Kode von jedem Zeichen, das im Programmlisting vorkommt, gespeichert.
Bei Verwendung des SAVE-Befehls werden die im Programmlisting vorkommenden Befehlsworte durch ein Zeichen abgekürzt. Wenn im Programmlisting zum Beispiel der INPUT-Befehl auftaucht, dann wird an Stelle der fünf ATASCII-Zeichen nur ein einziges Zeichen, nämlich die Abkürzung für INPUT, gespeichert.
Mit Hilfe des SAVE-Befehls kann immer nur das ganze Programm, das sich momentan im Speicher befindet, auf der Diskette festgehalten werden, während bei Verwendung des LIST-Befehls sowohl das ganze als auch nur ein Teil des Programmes gespeichert werden kann. In diesem Fall müssen Sie die Zeilennummer der ersten und der letzten Zeile, die noch gespeichert werden soll, angeben. Mit Hilfe des folgenden Befehls werden beispielsweise nur die Programmzeilen mit Zeilennummern zwischen 10 und 50 auf die Diskette geschrieben:

```
LIST "D:DATEN.LST",10,50
```

Laden eines Programmes

Programme, die mit Hilfe des SAVE-Befehls gespeichert wurden, können nur mit Hilfe des LOAD-Befehls geladen werden. Genauso gehören auch der LIST- und der ENTER-Befehl zusammen. LOAD und ENTER sind wegen des unterschiedlichen Aufzeichnungsformat nicht untereinander austauschbar.
Wie beim Speichern muß auch beim Laden von Programmen ein Filename, der Zusatz des Filenamens und eine Laufwerksnummer angegeben werden. Letztere kann auch weggelassen werden. In diesem Fall wird das Programm von Laufwerk 1 geladen. Bei Ausführung des folgenden Befehls wird ein Programm, das mit Hilfe des SAVE-Befehls gespeichert wurde, von Laufwerk 2 geladen:

```
LOAD "D2:ADDRESS.BAS"
```

Ein weiteres Beispiel. Diesmal wurde das Programm unter Verwendung des LIST-Befehls gespeichert und befindet sich auf der in Laufwerk 1 eingelegten Diskette:

```
ENTER "D:SCHACH.BAS"
```

Die Filenamen müssen dieselben sein, die auch bei der Speicherung der Programme verwendet wurden. Vor dem eigentlichen Ladevorgang wird geprüft, ob es ein File mit dem angegebenen Namen auf der in dem angegebenen Laufwerk eingelegten Diskette überhaupt gibt. Wenn das nicht der Fall ist, erfolgt eine Fehlermeldung (ERROR 170). Sonst wird das Programm von der Diskette geladen und nach Beendigung des Ladevorganges READY auf den Bildschirm geschrieben.
Bei Verwendung des LOAD-Befehls wird ein eventuell gerade im Speicher vorhandenes Programm gelöscht und durch das neu zu ladende Programm ersetzt. Bei Ausführung des ENTER-Befehls werden hingegen das neue und das alte Programm vermischt. Wenn dabei im neuen Programm eine Zeile vorkommt, die dieselbe Zeilennummer wie eine Zeile des bereits im Speicher befindlichen Programmes trägt, dann wird die alte Zeile durch die neue ersetzt. Wenn Sie die Vermischung der beiden Programme verhindern wollen, geben Sie vor Ausführung des ENTER-Befehls NEW ein.

Der RUN-Befehl

Es wird sicher sehr oft vorkommen, daß Sie ein Programm laden und danach sofort starten wollen. Normalerweise müssen sie zu diesem Zweck zwei Befehle geben:

```
LOAD "D:UMSATZ.BAS"
READY
RUN
```

Aus diesen zwei Schritten können Sie einen machen, indem Sie den Filenamen direkt hinter einen RUN-Befehl schreiben. Für das obige Beispiel sieht das dann so aus:

```
RUN "D:UMSATZ.BAS"
```

Bei Ausführung dieses Befehls wird das Programm geladen und automatisch gestartet.

Verketten von Programmen

Auch wenn ein RUN-Befehl in einem Programm vorkommt, wird ein anderes Programm geladen und gestartet. In Kapitel 5 wurde ausführlich erklärt, wie diese Verkettung von Programmen beim Programmrecorder funktioniert. Mit der Diskettenstation funktioniert das Ganze noch besser.
Wir werden nun drei kleine Programme schreiben und auf Diskette speichern. Das erste lädt und startet das zweite und das zweite lädt und startet das dritte. Geben Sie das erste Programm ein und speichern Sie es auf Diskette:

```
NEW
READY
10 PRINT "ERSTES PROGRAMM"
20 RUN "D1:P2.BAS"
SAVE "D1:P1.BAS"
READY
```

Damit wäre das erste Programm im File P1.BAS auf der in Laufwerk 1 eingelegten Diskette gespeichert.
Geben Sie nun das zweite Programm ein und speichern Sie es ebenfalls:

```
10 PRINT "ZWEITES PROGRAMM"
20 RUN "D1:P3.BAS"
SAVE "D1:P2.BAS"
READY
```

Damit wären die ersten beiden Programme auf der in Laufwerk 1 eingelegten Diskette gespeichert. Nun fehlt nur noch das dritte Programm:

```
10 PRINT "DRITTES PROGRAMM"
20 END
SAVE "D1:P3.BAS"
READY
```

Auf der Diskette befinden sich nun drei miteinander verkettete Programme. Laden und starten Sie das erste Programm mit Hilfe des RUN-Befehls:

```
RUN "D1:P1.BAS"
ERSTES PROGRAMM
ZWEITES PROGRAMM
DRITTES PROGRAMM
READY
```

Wie Sie sehen, wird vom ersten Programm das zweite und vom zweiten das dritte geladen und gestartet. Dieser Vorgang läuft vollautomatisch ohne Ihr Zutun ab.
Miteinander verkettete Programme sehen für den Anwender wie ein einziges großes Programm aus. Beim Programmrecorder muß der Anwender jedesmal die RETURN-Taste betätigen, wenn ein neues Programm geladen werden soll (siehe Kapitel 5). Bei der Diskettenstation ist das nicht notwendig.
Einen Nachteil müssen Sie bei der Verkettung von Programmen jedoch in Kauf nehmen: bevor ein neues Programm geladen wird, werden alle Variablen gelöscht. In diesem neuen Programm können Sie also keine Eingaben oder Berechnungen eines vorhergehenden Programmes verwenden.

Zusammenstellung von Unterprogrammen

Jeder Programmierer entwickelt im Laufe der Zeit Unterprogramme, die in fast jedem Programm verwendet werden. In Kapitel 4 haben Sie einige solcher Unterprogramme kennengelernt (Bild 4-16 bis 4-21, 4-31, 4-33, 4-36, 4-37 und 4-38). Bei Benutzung dieser Unterprogramme können Sie viel Zeit sparen. Allerdings müssen Sie diese Programme irgendwie eingeben. Wenn Sie sie bei jeder Verwendung neu eintippen, haben Sie schon nicht mehr soviel Zeit gespart. Es gibt allerdings einen anderen Weg: eine auf Diskette gespeicherte Zusammenstellung von Unterprogrammen.
Der Aufbau und die Benutzung einer solchen Zusammenstellung ist denkbar einfach. Jedesmal, wenn Sie ein Unterprogramm zum ersten Mal eingeben, speichern Sie es mit Hilfe des LIST-Befehls auf Diskette. Wenn Sie dieses Unterprogramm später bei der Erstellung eines Programmes benötigen, laden Sie es unter Verwendung des ENTER-Befehls. Es wird dann zusätzlich zu dem Hauptprogramm in den Speicher geladen.

Die Variablentabelle

Sie werden sich vielleicht daran erinnern, daß ATARI-BASIC alle Variablen- und Feldnamen, die in einem Programm oder auch im Eingabemodus verwendet wurden, in eine Tabelle einträgt. Bei der Speicherung eines Programmes mit Hilfe des SAVE-Befehls wird diese Tabelle ebenfalls gespeichert und beim erneuten Laden unter Verwendung des LOAD-Befehls wieder geladen. Sie ersetzt dann die bisher im Speicher befindliche Tabelle. Bei Verwendung des LIST-Befehls zum Speichern eines Programmes wird die Variablentabelle nicht mitgespeichert. Folglich kann beim Laden eines Programmes mit Hilfe des ENTER-Befehls auch keine Tabelle geladen werden. Die bisher im Speicher befindliche Tabelle bleibt solange erhalten, bis Sie einen NEW-Befehl geben.
Nachdem Sie eine gewisse Zeit mit einem Programm gearbeitet haben, werden sich in der Variablentabelle Variablennamen ansammeln, die Sie überhaupt nicht mehr benötigen. Um diese Namen zu löschen, speichern Sie zunächst das Programm mit Hilfe des LIST-Befehls und löschen anschließend die Tabelle durch Eingabe von NEW. Dann laden Sie das dabei ebenfalls gelöschte Programm unter Verwendung des ENTER-Befehls.

Datenfiles auf der Diskette

Die Diskettenstation ist das ideale Speichermedium für große Datenmengen. Mit Hilfe der BASIC-Befehle PRINT # und PUT können Sie Daten auf einer Diskette ablegen und mit INPUT # und GET wieder laden.

Datenfiles, Records und Fields

Für den Computer sind Datenfiles und Files, in denen Programme gespeichert wurden, absolut gleich. Er sieht in beiden nur Zahlen. Ein Unterschied ergibt sich erst bei einer unterschiedlichen Interpretation dieser Zahlen. Für den Anwender sind in einem Programmfile Programmzeilen und in einem Datenfile numerische und Stringvariablen gespeichert. Alle Files sind nach einem gewissen Schema aufgebaut. In einem File könnte zum Beispiel ein Adressenverzeichnis gespeichert sein, also eine Zusammenstellung von Namen und Adressen. Ein Name und die dazugehörige Adresse bilden dann einen Record. In einem Record sind mehrere Dinge gespeichert: der Name, die Straße, die Postleitzahl und der Wohnort. Das sind die Fields. Normalerweise sind in jedem Record eines Files die gleichen Fields enthalten. Nur der Inhalt der Fields ist unterschiedlich.

Die zwei verschiedenen Arten des Diskettenzugriffs

Es kann auf zwei verschiedene Arten auf die Diskette zugegriffen werden. Die erste Möglichkeit ist der sequentielle Zugriff (SEQUENTIAL ACCESS). Ein sequentielles File ist genauso aufgebaut wie ein Kassettenfile. Wenn Sie das letzte Field eines solchen Files lesen oder beschreiben wollen, dann müssen Sie alle davor liegenden Fields lesen oder beschreiben. Nur für wenige Anwendungen verwendet man den sequentiellen Zugriff.
Mehr Möglichkeiten bietet der wahlfreie Zugriff (RANDOM ACCESS). Jeder Record eines Files kann genauso einfach gelesen oder beschrieben werden, ganz gleich, um welchen Record es sich handelt. Für die meisten Anwendungen wird diese Zugriffsart verwendet.
Unter DOS 2.0S kann sowohl mit dem sequentiellen als auch mit dem wahlfreien Zugriff gearbeitet werden, während DOS 1.0 nur die sequentielle Zugriffsart kennt.

Die Datenspeicherung

Um unter Tausenden von Zeichen, die auf einer Diskette gespeichert werden können, ein ganz bestimmtes schnell finden zu können, unterteilt das Disk Operating System die Diskette in 720 Teile, die Sektoren. In jedem Sektor können 128 Zeichen gespeichert werden. DOS 2.0S benötigt 13 Sektoren von jeder Diskette für interne Zwecke. In den Sektoren 1, 2 und 3 befindet sich das Programm, mit dessen Hilfe das Disk Operating System gebootet wird. Auf den Sektoren 361 bis 368 befindet sich das Inhaltsverzeichnis der Diskette. In Sektor 360 wird festgehalten, welche Sektoren einer Diskette belegt und welche frei sind. Außerdem ist der letzte Sektor einer Diskette (Sektor 720) nicht beschreibbar. Unter DOS 2.0S stehen somit also noch 707 Sektoren zur freien Verfügung.
Unter DOS 1.0 wird nur Sektor 1 für das BOOT-Programm benötigt. Damit stehen zwei weitere Sektoren für die Datenspeicherung bereit. Ansonsten ist die Belegung der Sektoren genauso wie bei DOS 2.0S.

Die Spuren

Um die Suche nach einem bestimmten Sektor zu vereinfachen, wird die Diskette in 40 Spuren in Form von konzentrischen Kreisen unterteilt. Auf jeder Spur befinden sich 18

Sektoren (Bild 7-10). Da der Schreib-Lese-Kopf auf einer bestimmten Spur positioniert werden kann, müssen maximal nur 18 Sektoren gelesen werden, bevor der gewünschte gefunden wurde.

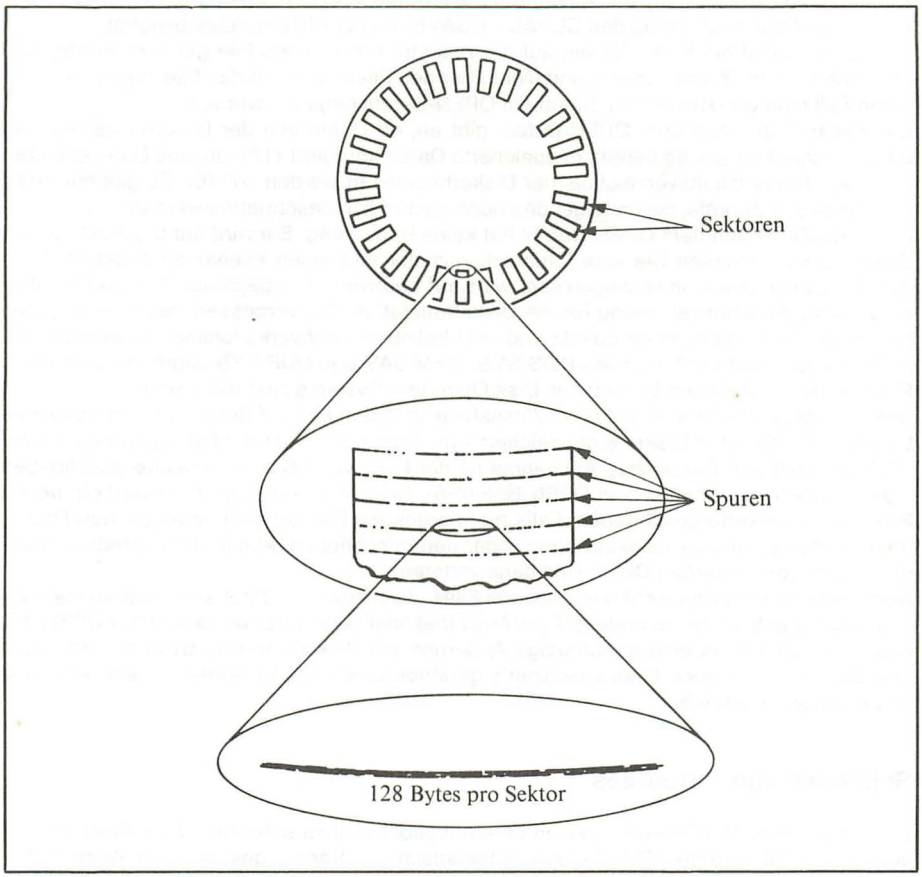


Bild 7-10: So ist eine Diskette unterteilt

Öffnen von Datenfiles

Bevor auf ein File auf der Diskette zugegriffen werden kann, muß es geöffnet werden. Das Disk Operating System informiert Sie daraufhin darüber, ob und wo sich das File auf der Diskette befindet. Außerdem wird Speicherplatz für einen Filebuffer reserviert. Die Funktion des Filebuffers entspricht der des Diskettenbuffers. Mit seiner Hilfe kann auf einen kleinen Teil des Files, für das er zuständig ist, zugegriffen werden, ohne daß die Diskettenstation akiviert werden muß. Damit wird viel Zeit gespart.
Mit Hilfe des OPEN-Befehls kann ein Ein-/Ausgabekanal für ein bestimmtes File einer Diskette geöffnet werden, zum Beispiel so:

```
OPEN #2,8,0,"D1:FILENAME.EXT"
```

Bei Ausführung dieses Befehls wird Kanal 2 für Ausgabe auf das File FILENAME.EXT auf

der in Laufwerk 1 eingelegten Diskette geöffnet.
Die erste Zahl nach dem OPEN-Befehl ist die Kanalnummer. Im BASIC stehen die Kanäle 1 bis 5 und unter bestimmten Bedingungen auch 6 und 7 zur Verfügung (siehe Kapitel 4). Kanal 6 wird nämlich auch zur Ausführung der Grafikbefehle (siehe auch Kapitel 8 und 9) und Kanal 7 zur Ausführung des CLOAD-, CSAVE- und LPRINT-Befehls benötigt.
Nachdem einmal ein Kanal für ein auf der Diskette befindliches File geöffnet wurde, wird nur noch über die Kanalnummer und nicht über den Filenamen auf das File zugegriffen. Auf jeden Fall muß vor dem ersten Zugriff ein OPEN-Befehl gegeben werden.
Die zweite Zahl nach dem OPEN-Befehl gibt an, ob Daten von der Diskette gelesen (4) oder geschrieben (8), ob bereits gespeicherte Daten korrigiert (12), ob eine Datei erweitert (9) oder ob das Inhaltsverzeichnis der Diskette gelesen werden soll (6). Es gibt nur diese fünf Arten des Zugriffs, die im folgenden noch ausführlich beschrieben werden.
Die dritte Zahl nach dem OPEN-Befehl hat keine Bedeutung. Sie wird auf 0 gesetzt. Außer diesen Zahlen müssen Sie eine Laufwerksnummer und einen Filenamen angeben. Falls Sie die Laufwerksnummer weglassen, wird auf Laufwerk 1 zugegriffen. Sie dürfen allerdings nicht die Kennzeichnung für die Diskettenstation (D:) vergessen. Sie können jeden beliebigen Filenamen, jeden Zusatz und jede beliebige Laufwerksnummer verwenden. Sie sollten jedoch nicht auf die Files DOS.SYS, MEM.SAV und DUP.SYS zugreifen, weil diese Files für den fehlerfreien Betrieb des Disk Operating Systems zuständig sind.
Das angegebene File muß, mit einer Ausnahme, in jedem Fall auf der in dem angegebenen Laufwerk eingelegten Diskette gespeichert sein. Wenn das nicht der Fall ist, tritt ein Fehler (ERROR 170) auf. Die einzige Ausnahme ist der Fall, wenn Sie die Diskette beschreiben wollen (zweite Zahl hinter dem OPEN-Befehl=8). Nur bei dieser Zugriffsart wird ein neues File auf die Diskette geschrieben. Falls hier bereits ein File mit dem angegebenen Filenamen existierte, wird es gelöscht und wieder neu geschrieben. Alle in dem bereits vorhandenen File gespeicherten Daten sind dann verloren.
Normalerweise können nicht mehr als drei Files gleichzeitig geöffnet sein. Selbstverständlich muß für jedes File ein anderer Ein-/Ausgabekanal verwendet werden. Unter DOS 2.0S kann man allerdings eine geringfügige Änderung am Disk Operating System vornehmen und dann bis zu sieben Files gleichzeitig geöffnet haben. Diese Änderung wird am Ende des Kapitels beschrieben.

Schließen von Datenfiles

Es gibt mehrere Möglichkeiten, einen Ein-/Ausgabekanal zu schließen. Zum Beispiel werden bei Ausführung des END-Befehls automatisch alle Kanäle geschlossen. Auch im folgenden Programm wird der in Zeile 200 geöffnete Ausgabekanal mit Hilfe des END-Befehls in Zeile 400 geschlossen:

```
100 PRINT "DAS FILE WIRD GEOEFFNET..."
200 OPEN #1,8,0,"D1:DATFILE.TMP"
300 PRINT "DAS FILE IST OFFEN!!"
400 END
```

Dieses Programm hat einen Fehler: man kann nicht auf den ersten Blick sehen, ob und wo das File geschlossen wird. Aus diesem Grund ist es vorteilhafter, wenn Sie zum Schließen eines Files den CLOSE-Befehl verwenden. Besonders wichtig ist das, falls auf die Diskette geschrieben wurde. Wenn in diesem Fall ein File nicht geschlossen wird, können Daten verloren gehen oder sogar Daten auf einer anderen Diskette gelöscht werden.

Verbessern Sie das obige Programm, indem Sie eine zusätzliche Zeile (Zeile 390) einfügen:

```
100 PRINT "DAS FILE WIRD GEOEFFNET..."
200 OPEN #1,8,0,"D1:DATFILE.TMP"
300 PRINT "DAS FILE IST OFFEN!!"
390 CLOSE #1
400 END
```

Beschreiben von Datenfiles

Es genügt schließlich nicht, ein File nur zu öffnen und zu schließen, denn Sie wollen ja irgend etwas in diesem File speichern. In diesem Abschnitt wird das obige Programm so geändert, daß damit Daten auf der Diskette gespeichert werden können.
Genau wie Sie Daten an den Programmrecorder oder den Drucker übertragen, machen Sie es auch bei der Diskettenstation: mit Hilfe des PRINT #-Befehls. Alles, was gedruckt werden kann, kann damit auch in einem Diskettenfile gespeichert werden. In der Tat können Sie ein sequentielles File mit einem Stück Druckerpapier vergleichen. Der Drucker speichert jedes empfangene Zeichen in seinem Buffer und druckt den Inhalt dieses Buffers erst aus, wenn er voll ist. Entsprechend werden die Daten, die Sie in einem Diskettenfile speichern wollen, erst im Filebuffer zwischengespeichert. Der Inhalt dieses Filebuffers wird erst dann auf die Diskette geschrieben, wenn er voll ist.
Damit die mit Hilfe des PRINT #-Befehls übertragenen Daten auch tatsächlich in dem gewünschten File gespeichert werden, geben Sie die Kanalnummer an, die Sie auch beim Öffnen des Files mit Hilfe des OPEN-Befehls verwendet haben. Im letzten Beispiel wurde dem File DATFILE.TMP Kanal 1 zugeordnet. Mit Hilfe des folgenden Programmes wird dieses File beschrieben:

```
100 PRINT "DAS FILE WIRD GEOEFFNET..."
200 OPEN #1,8,0,"D1:DATFILE.TMP"
300 PRINT "DAS FILE IST OFFEN!!"
310 PRINT #1;"EIN MAENNLEIN STEHT IM W
ALDE"
320 PRINT #1;"GANZ STILL UND STUMM"
390 CLOSE #1
400 END
```

Wenn Sie dieses Programm starten, wird das File DATFILE.TMP zunächst gelöscht, falls es sich bereits auf der Diskette befand, und dann neu geschrieben und geöffnet, denn mit Hilfe der zweiten Zahl nach dem OPEN-Befehl (8) in Zeile 200 wird festgelegt, daß auf die Diskette geschrieben werden soll. In dem neuen File sind zu diesem Zeitpunkt noch keine Daten gespeichert. Der OPEN-Befehl in Zeile 200 sorgt außerdem dafür, daß der Bufferzeiger auf den Anfang des Filebuffers zeigt. Der Filebuffer sieht zu diesem Zeitpunkt wie in Bild 7-11 gezeigt aus.

Mit Hilfe des ersten PRINT #-Befehls in Zeile 310 werden 28 Zeichen gespeichert, die allerdings zunächst in den Filebuffer gelangen. Der Bufferzeiger befindet sich dann an der 29. Stelle des Buffers. Da der PRINT #-Befehl weder mit einem Komma noch mit einem Semikolon endet, wird zusätzlich ein EOL-Zeichen im Filebuffer gespeichert. Damit rutscht der Bufferzeiger an die 30. Stelle. Der Buffer sieht jetzt wie in Bild 7-12 gezeigt aus.
Mit Hilfe des zweiten PRINT #-Befehls in Zeile 320 gelangen noch einmal 20 Zeichen und ein EOL-Zeichen in den Filebuffer. Damit steht der Zeiger des Filebuffers an der 51. Stelle. Bild 7-13 zeigt, wie der Buffer jetzt aussieht. Zwei Fields sind in ihm gespeichert. Am Ende von jedem Field steht ein EOL-Zeichen.

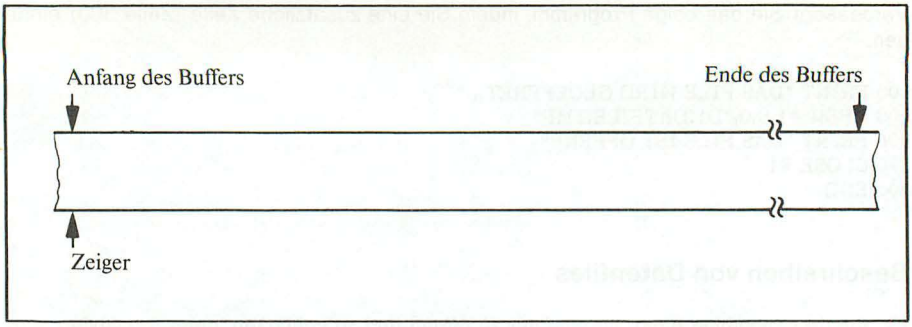


Bild 7-11: Ein leerer Filebuffer

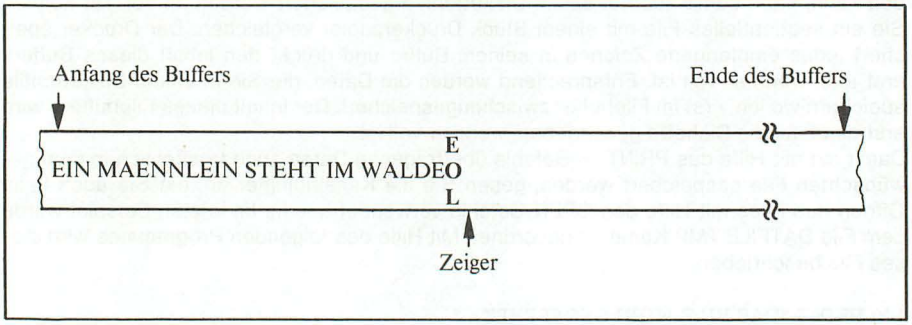


Bild 7-12: Teilweise beschriebener Filebuffer

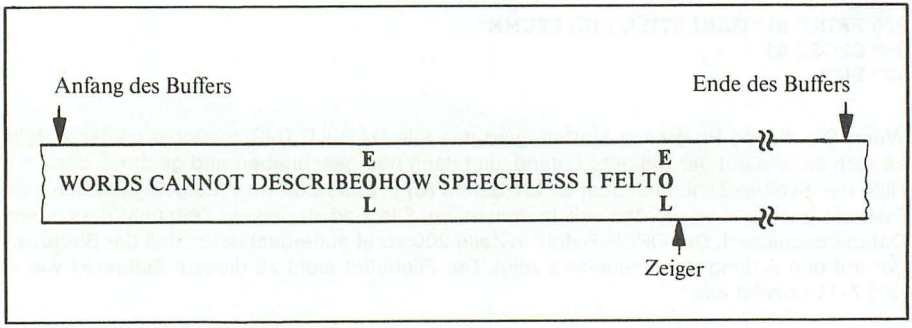


Bild 7-13: Filebuffer mit zwei Fields

Erst mit der Ausführung des CLOSE-Befehls in Zeile 390 wird der Inhalt des Filebuffers auf der Diskette gespeichert. Da Kanal 1 für das File DATENFILE.TMP geöffnet wurde, erfolgt auch die Speicherung in diesem File.
Im Filebuffer können 125 Zeichen gespeichert werden. Eigentlich ist sogar Platz für 128 Zeichen vorhanden, aber drei davon werden für interne Zwecke benötigt. Sobald der Filebuffer voll ist, wird er auf der Diskette gespeichert.

Es gibt nicht nur für den Filebuffer einen Zeiger, sondern auch für das File selbst. Bei einem sequentiellen File kann sich dieser Zeiger nur in eine Richtung bewegen, nämlich zum Fileende. Wenn bei einem gewissen Stand dieses Zeigers auf weiter in Richtung Fileanfang liegende Daten zugegriffen werden soll, muß das File geschlossen und wieder geöffnet werden. Danach befindet sich der Zeiger am Fileanfang.
Experimentieren Sie ein wenig mit dem obigen Programm. Verändern Sie zum Beispiel die Daten, die mit Hilfe des PRINT #-Befehls gespeichert werden oder versuchen Sie, stattdessen numerische Variablen, Stringvariablen oder Konstanten zu speichern. Zwischen dem OPEN- und dem CLOSE-Befehl können Sie beliebig viele PRINT #-Befehle zusätzlich einfügen.
Wenn sich am Ende eines PRINT #-Befehls ein Komma oder ein Semikolon befindet, wird kein EOL-Zeichen erzeugt. Die Daten eines darauffolgenden PRINT #-Befehls werden im selben Field gespeichert. Ein Beispiel dafür ist das folgende Programm:

```
100 OPEN #1,8,0,"D:EINFIELD.TMP"
200 PRINT #1;"DAS KOMMT ZUERST";
300 PRINT #1;"UND DANACH DAS"
400 CLOSE #1
500 END
```

Der Text des zweiten PRINT #-Befehls (Zeile 300) wird direkt im Anschluß an den Text des ersten PRINT #-Befehls ohne Einfügung von irgendeinem anderen Zeichen gespeichert. In dem angegebenen File ist daher nur ein Field gespeichert (Bild 7-14).

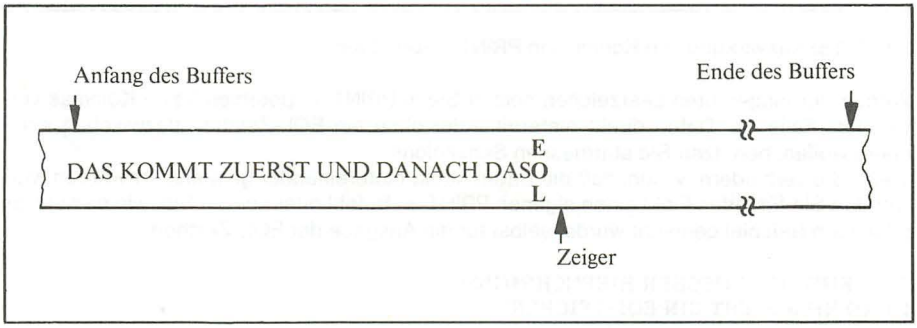


Bild 7-14: Auswirkung eines Semikolons am Ende eines PRINT #-Befehls

Die Verwendung von Kommas in PRINT #-Befehlen

Im Zusammenhang mit PRINT #-Befehlen werden Kommas häufig zur Trennung verwendet:

```
745 PRINT #4;"HALLO!","WIE LAUTET IHR
NAME?"
```

Sie können allerdings genausogut am Ende eines PRINT #-Befehls auftauchen:

```
9456 PRINT #2;"ENDE",
```

Für den Computer macht es keinen Unterschied, ob Daten auf den Bildschirm oder eine Diskette geschrieben werden. Beim Bildschirm wird der Cursor mit Hilfe des Kommas auf den nächsten Tabulatorstop gesetzt. Außerdem wird die Ausgabe eines EOL-Zeichens

verhindert. In Kapitel 4 sind die Auswirkungen des Kommas sehr ausführlich behandelt worden. Bei einem Diskettenfile tritt der Zeiger des Files an die Stelle des Cursors. Sehen wir uns den folgenden Befehl einmal etwas genauer an:

```
PRINT #1;1,2,3,
```

Wenn Kanal 1 für Ausgabe auf ein Diskettenfile geöffnet ist, dann wird mit Hilfe dieses Befehls ein Field in diesem File gespeichert. Da die zu speichernden Daten durch Kommas getrennt sind, folgen auf jede Zahl neun Leerzeichen. Ein EOL-Zeichen wird nicht ausgegeben, weil der Befehl auch mit einem Komma endet. Nach der Ausführung sieht der Filebuffer wie in Bild 7-15 gezeigt aus.

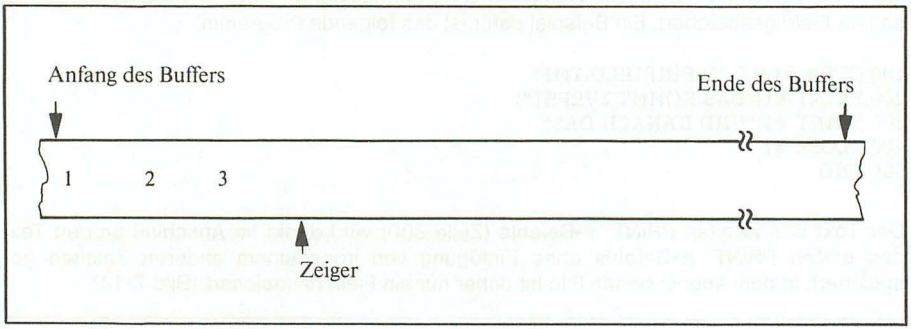


Bild 7-15: Auswirkung von Kommas in PRINT #-Befehlen

Wegen der eingefügten Leerzeichen sollten Sie in PRINT #-Befehlen keine Kommas verwenden. Falls Sie Daten direkt hintereinander ohne ein EOL-Zeichen dazwischen speichern wollen, benutzen Sie stattdessen Semikolons.
Wenn Sie verhindern wollen, daß die Daten direkt hintereinander gespeichert werden, verwenden Sie für jedes Field einen eigenen PRINT #-Befehl oder sorgen Sie, wie es auch im folgenden Beispiel gemacht wurde, selbst für die Ausgabe der EOL-Zeichen:

```
233 PRINT #5;"ZWISCHEN HIER";CHR$(155)
;"UND HIER STEHT EIN EOL-ZEICHEN"
```

CHR$(155) entspricht dem EOL-Zeichen. Falls Sie diese Methode mehrmals anwenden wollen, ist es vorteilhafter, das EOL-Zeichen in einer Stringvariablen zu speichern:

```
 5 DIM R$(1)
 10 R$=CHR$(155):REM EOL
    .
    .
    .
431 PRINT #2;NAME$;R$;BERUF$;R$;ALTER$
```

Beschreiben von Datenfiles mit Hilfe des PUT-Befehls

Mit Hilfe des PUT-Befehls kann eine Zahl mit einem Wert zwischen 0 und 255 auf ein Diskettenfile geschrieben werden. Normalerweise steht diese Zahl für ein ATASCII-Zeichen. Jede Zahl belegt im File genausoviel Platz wie ein Zeichen, das mit Hilfe des PRINT #-Befehls gespeichert wurde. Bei Verwendung des PUT-Befehls werden keine EOL-Zeichen

geschrieben. Das folgende Programm schreibt einen in Anführungszeichen gesetzten Text ("HALLO") und ein EOL-Zeichen auf das angegebene File:

```
10 OPEN #5,8,0,"D:HALLO.TXT"
20 PUT #5,34:REM "
30 PUT #5,72:REM H
40 PUT #5,65:REM A
50 PUT #5,76:REM L
60 PUT #5,76:REM L
70 PUT #5,79:REM O
80 PUT #5,34:REM "
90 PUT #5,155:REM (EOL)
100 CLOSE #5
110 END
```

Lesen von sequentiellen Datenfiles

Die in einem Diskettenfile gespeicherten Daten können natürlich auch wieder abgerufen werden. Sie werden mit Hilfe des INPUT #- und des GET-Befehls geladen und Variablen zugewiesen. Um die Funktion dieser Befehle zu veranschaulichen, geben Sie zunächst das folgende Programm ein und starten Sie es. Damit wird ein File mit dem Namen DATENFIL.TMP erzeugt:

```
200 OPEN #1,8,0,"D1:DATENFIL.TMP"
310 PRINT #1;"MAN KANN SPEICHERN"
320 PRINT #1;"UND WIEDER ABRUFEN"
390 CLOSE #1
400 END
```

Mit Hilfe des folgenden Programmes werden die in DATENFIL.TMP gespeicherten Daten gelesen und auf dem Bildschirm zur Anzeige gebracht:

```
100 DIM A$(100)
190 REM File zum Lesen oeffnen
200 OPEN #1,4,0,"D:DATENFIL.TMP"
300 INPUT #1;A$
400 PRINT A$
500 GOTO 300
600 CLOSE #1
700 END
```

Die Ausgabe dieses Programmes sieht folgendermaßen aus:

```
MAN KANN SPEICHERN
UND WIEDER ABRUFEN

ERROR-  136 AT LINE 300
```

Der Fehler trat auf, weil versucht wurde, mehr Daten zu lesen als in dem File vorhanden waren. Das Programm wurde in Zeile 300 gestoppt. Der CLOSE-Befehl zum Schließen des Files (Zeile 600) konnte also nicht mehr ausgeführt werden. Das ist in diesem Fall aber nicht schlimm, weil das File gelesen wurde und somit keine Daten im Filebuffer standen, die noch auf die Diskette zu schreiben gewesen wären.
Es ist keine besonders elegante Lösung, wenn ein Programm durch einen Fehler beendet wird. Sie können diesen Fehler jedoch mit Hilfe des TRAP-Befehls abfangen. Sie müssen

nur eine Zeile (Zeile 210) in das obige Programm einfügen:

```
100 DIM A$(100)
190 REM File zum Lesen oeffnen
200 OPEN #1,4,0,"D:DATENFIL.TMP"
210 TRAP 600
300 INPUT #1;A$
400 PRINT A$
500 GOTO 300
600 CLOSE #1
700 END
```

Das ist eine saubere Lösung, bei der keine Fehlermeldung mehr erfolgt.
Zur Sicherheit sollte allerdings noch überprüft werden, ob es sich bei dem aufgetretenen Fehler auch tatsächlich um einen end-of-file-Error handelt, denn sonst würde jeder Fehler, der überhaupt auftreten kann, wie ein end-of-file-Error behandelt. Damit diese Überprüfung erfolgt, ändern sie das obige Programm folgendermaßen:

```
100 DIM A$(100)
190 REM File zum Lesen oeffnen
200 OPEN #1,4,0,"D:DATENFIL.TMP"
210 TRAP 510
300 INPUT #1;A$
400 PRINT A$
500 GOTO 300
505 REM Nummer des Errors holen
510 ERR=PEEK(195)
515 REM Ende des Files?
520 IF ERR=135 THEN 600
525 REM Sonst die Nummer des Errors au
sgeben
530 PRINT "ERROR ";ERR;" IST AUFGETRET
EN!"
600 CLOSE #1
700 END
```

Mit Hilfe des INPUT #-Befehls wird ein Field eines Datenfiles, das heißt also, alle Zeichen bis zum Auftreten eines EOL, gelesen. Die in diesem Field gespeicherten Daten werden dann der angegebenen Variablen zugewiesen. Die Daten und die Variable müssen allerdings zueinander passen. Sollte zum Beispiel einmal einer numerischen Variablen ein String zugewiesen werden, dann tritt dabei ein Fehler auf.

Lesen von Files mit Hilfe des GET-Befehls

Mit Hilfe des GET-Befehls wird ein einzelnes Zeichen eines Datenfiles in Form einer Zahl mit einem Wert zwischen 0 und 255 gelesen. Wie diese Zahlen interpretiert werden, ist durch das Programm festgelegt. In den meisten Fällen werden diese Zahlen unter Verwendung der CHR$-Funktion in ATASCII-Zeichen umgewandelt. Das folgende Programm liest dasselbe File wie das vorhergehende, allerdings mit Hilfe der CHR$-Funktion:

```
190 REM File zum Lesen oeffnen
200 OPEN #1,4,0,"D:DATENFIL.TMP"
210 TRAP 510
300 GET #1,A
400 PRINT CHR$(A);
```

```
500 GOTO 300
505 REM Nummer des Errors holen
510 ERR=PEEK(195)
515 REM Ende des Files?
520 IF ERR=135 THEN 600
525 REM Sonst die Nummer des Errors au
sgeben
530 PRINT "ERROR ";ERR;" IST AUFGETRET
EN!"
600 CLOSE #1
700 END
```

Erweiterung eines bereits bestehenden Files

Wenn Sie Daten auf einer Diskette speichern wollen und zu diesem Zweck ein File öffnen (zweite Zahl nach dem OPEN-Befehl=8), dann wird, falls ein File mit dem angegebenen Namen bereits existiert, dieses zunächst vollständig gelöscht und dann der Zeiger des neuen Files auf den Fileanfang gesetzt. Es ist jedoch auch möglich, ein bereits bestehendes File zu erweitern.
In diesem Fall muß die zweite Zahl hinter dem OPEN-Befehl eine 9 sein. Der Zeiger des Files wird dann auf das Fileende gesetzt. Falls das angegebene File nicht existiert, tritt ein Fehler (ERROR 170) auf.
Nehmen Sie zum Beispiel an, daß sich auf der Diskette ein File mit dem Namen ANFUEG.TXT befindet. In diesem File sind die Worte "ES WAR EINMAL" und ein EOL-Zeichen gespeichert. Nun wird das folgende Programm ausgeführt:

```
210 OPEN #2,9,0,"D:ANFUEG.TXT"
220 PRINT #2;"EINE DISKETTENSTATION."
230 PRINT #2;"DIE HATTE KEINE DISKETTE
."
240 CLOSE #2
250 END
```

Mit Hilfe des OPEN-Befehls in Zeile 210 wird festgelegt, daß das File erweitert werden soll. Der Zeiger des Files wird demzufolge auf das Fileende gesetzt (Bild 7-16). Die Daten, die nun zusätzlich in dem File gespeichert werden, werden dort angefügt (Bild 7-17).

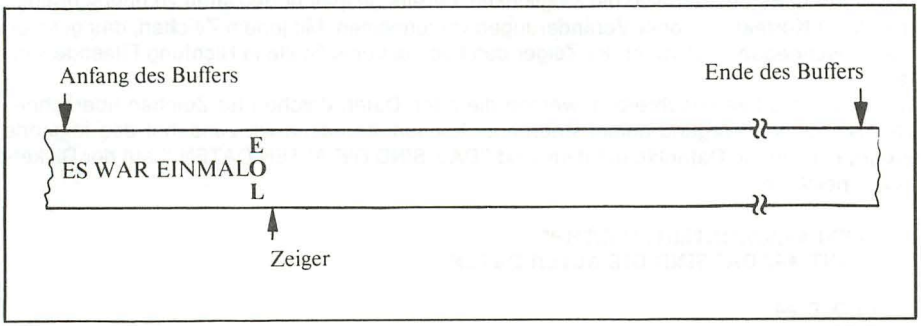


Bild 7-16: Ein zur Erweiterung geöffnetes File

Jedesmal, wenn Sie einen OPEN-Befehl zur Erweiterung eines Files geben, wird automatisch ein neuer Sektor (128 Zeichen Speicherkapazität) für dieses File reserviert, auch dann, wenn noch Platz im letzten Sektor, in dem das File bis zu diesem Zeitpunkt gespei-

chert war, vorhanden ist. Wenn ein Programm nicht optimal ausgelegt ist, kann aufgrund dieser Tatsache sehr schnell sehr viel Speicherplatz einer Diskette verbraucht werden. Diesen Nachteil weist der im nächsten Abschnitt behandelte zweite Weg zur Erweiterung eines Files nicht auf.

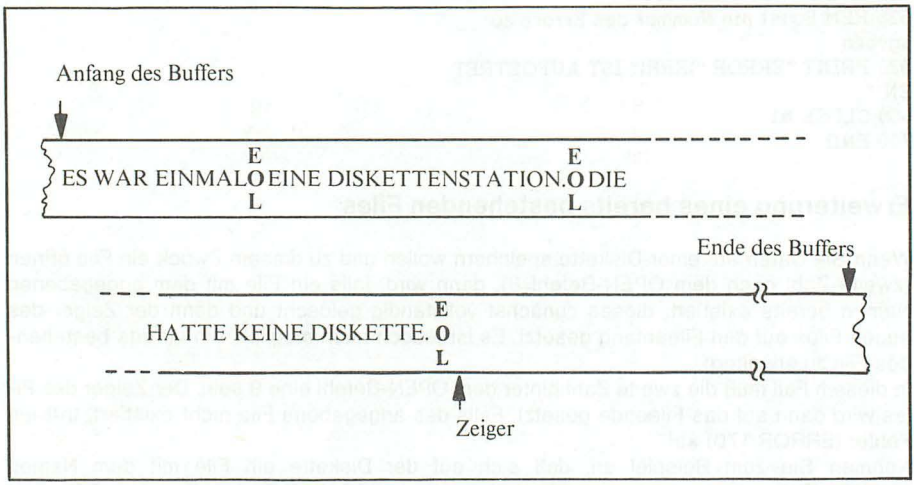


Bild 7-17: Erweiterung eines bereits bestehenden Files

Korrigieren oder Verändern der in einem Datenfile gespeicherten Daten

Wenn Sie als zweite Zahl nach einem OPEN-Befehl eine 8 oder eine 9 angeben, können Sie nur auf die Diskette schreiben. Bei Angabe einer 4 können Sie nur lesen. Falls Sie lesen und schreiben wollen, müssen Sie eine 12 einsetzen. In diesem Fall muß ein File mit dem auf den OPEN-Befehl folgenden Filenamen bereits existieren.
Nachdem Sie ein File mit einem solchen OPEN-Befehl geöffnet haben, befindet sich der Zeiger des Files am Fileanfang. Es können nun sowohl Daten gelesen als auch geschrieben werden. Es besteht also die Möglichkeit, bereits gespeicherte Daten zu überschreiben und damit Korrekturen oder Veränderungen vorzunehmen. Mit jedem Zeichen, das gelesen oder geschrieben wird, rückt der Zeiger des Files um eine Stelle in Richtung Fileende weiter.
Wenn Sie das File beschreiben, werden die alten Daten Zeichen für Zeichen überschrieben. Um diesen Vorgang demonstrieren zu können, benutzen wir zunächst das folgende Programm, um ein Datenfile mit dem Text "DAS SIND DIE ALTEN DATEN " auf der Diskette zu speichern:

```
100 OPEN #4,8,0,"D:TESTFILE.TMP"
110 PRINT #4;"DAS SIND DIE ALTEN DATEN
"
120 CLOSE #4
130 END
```

Mit Hilfe des folgenden Programmes wird der gespeicherte Text verändert:

```
200 OPEN #3,12,0,"D:TESTFILE.TMP"
210 PRINT #3;"HALLO"
220 CLOSE #3
230 END
```

Nach der Ausführung des OPEN-Befehls in Zeile 200 befindet sich der Zeiger des Files am Fileanfang. Der mit Hilfe des PRINT #-Befehls in Zeile 210 ausgegebene Text wird folglich auch dort gespeichert. Die ersten sechs Zeichen des alten Files werden durch "HALLO" und ein EOL-Zeichen ersetzt. In dem geänderten File befinden sich jetzt zwei, durch EOL-Zeichen voneinander getrennte Fields (Bild 7-18).

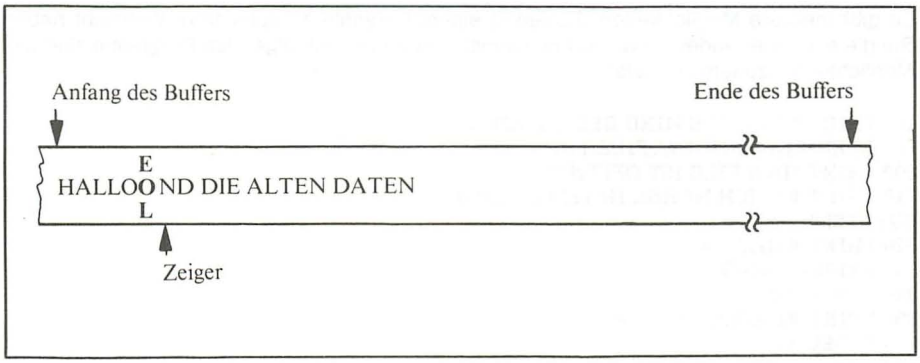


Bild 7-18: Überschreiben von bereits gespeicherten Daten

Wenn Sie Daten aus dem alten File lesen, bewegt sich dabei der Zeiger des Files in Richtung Fileende. So können Sie an jedem beliebigen Punkt des Files Daten verändern oder korrigieren. Sie lesen einfach solange, bis sich der Zeiger an der gewünschten Stelle befindet und schreiben dann die neuen Daten in das File. Lesen können Sie sowohl mit Hilfe des INPUT #- als auch mit dem GET-Befehl. Mit INPUT # wird ein ganzes Field und mit GET ein einzelnes Zeichen gelesen. Im folgenden Beispiel wird der GET-Befehl verwendet:

```
100 TRAP 1900:REM Falls das File nicht
 existiert
190 REM File oeffnen, um gespeicherte
Daten zu veraendern
200 OPEN #3,12,0,"D:WORTE.TXT"
600 TRAP 1700:REM Falls im File kein $
 gespeichert ist
790 REM Zeichen lesen, bis $ gefunden
wurde
800 GET #3,A:IF CHR$(A)<>"$" THEN 800
900 REM $ wurde gefunden
1000 PRINT #3;"1234.56";
1100 REM Das Semikolon am Ende des obi
gen Befehls verhindert, dass ein EOL-
1200 REM Zeichen gespeichert wird
1300 CLOSE #3
1400 PRINT "DATEN VERAENDERT"
1500 GOTO 2000
1700 PRINT "KEIN DOLLAR-ZEICHEN IM FIL
E GEFUNDEN"
```

```
1800 GOTO 2000
1900 PRINT "FILE EXISTIERT NICHT"
2000 END
```

In Zeile 200 wird ein File geöffnet, um die darin gespeicherten Daten zu verändern. Falls das File nicht existiert, wird das Programm beendet (Zeile 100 und 1900). Dann wird das File Zeichen für Zeichen gelesen, bis ein Dollar-Zeichen auftritt (Zeile 800). Schließlich werden die auf dieses Dollar-Zeichen folgenden sieben Zeichen durch die Zahl 1234.56 ersetzt (Zeile 1000).

Die Speicherung von Zahlen in Datenfiles

Es gibt mehrere Möglichkeiten, Zahlen in einem Datenfile zu speichern. Vielleicht haben Sie die eine oder andere sogar schon einmal ausprobiert. Im folgenden Programm sind alle Möglichkeiten zusammengefaßt:

```
100 PRINT "DAS FILE WIRD GEOEFFNET..."
200 OPEN #1,8,0,"D1:DATFILE.TMP"
300 PRINT "DAS FILE IST OFFEN!!"
310 PRINT #1;"ICH WOHNE IN 1234 DOESEN
BUETTEL"
320 PRINT #1;1,2,3,4,5
330 A=10:B=20:C=30
340 D=40:E=50
350 PRINT #1;A,B,C,D+A,E*B
390 CLOSE #1
400 END
```

Wie Sie sehen, kann man eine Zahl sowohl indirekt als Teil einer Stringvariablen (Zeile 310) als auch direkt in Form einer Konstanten (Zeile 320) oder einer numerischen Variablen beziehungsweise eines arithmetischen Ausdrucks (Zeile 350) speichern.
Wenn Sie Zahlen direkt speichern, können Sie sie mit Hilfe des INPUT #-Befehls wieder lesen. Das folgende Programm ist ein Beispiel dafür, wie das gemacht werden sollte:

```
95 REM File zum Speichern der Daten oe
ffnen
100 OPEN #2,8,0,"D:ZAHLEN.DAT"
105 REM 10 Zahlen eingeben
110 FOR J=1 TO 10
120 PRINT "GEBEN SIE EINE ZAHL EIN, DI
E GESPEI-"
125 PRINT "CHERT WERDEN SOLL";
130 INPUT N
135 REM Die eingegebene Zahl im File a
bspeichern
140 PRINT #2;N
150 NEXT J
160 CLOSE #2
195 REM File zum Lesen der Daten oeffn
en
200 OPEN #2,4,0,"D:ZAHLEN.DAT"
205 REM 10 Zahlen lesen
210 PRINT "SIE HABEN DIE FOLGENDEN ZAH
LEN EIN-"
215 PRINT "GEGEBEN:"
220 FOR I=1 TO 10
230 INPUT #2,N
235 REM Die geladene Zahl anzeigen
240 PRINT N
250 NEXT I
260 CLOSE #2
300 END
```

Zunächst wird auf der Diskette ein neues File geöffnet (Zeile 100). Dann müssen Sie zehn Zahlen eingeben. Jede Zahl wird nach der Eingabe gespeichert (Zeile 110 bis 160). Bei der Ausführung des CLOSE-Befehls (Zeile 160) wird die Diskettenstation aktiviert, um die noch im Filebuffer stehenden Daten auf die Diskette zu schreiben. Schließlich wird das soeben geschriebene File wieder geöffnet, um die Daten zu lesen (Zeile 200). Die nacheinander geladenen Zahlen werden auf dem Bildschirm ausgegeben (Zeile 220 bis 250).
Zum besseren Verständnis soll hier kurz beschrieben werden, wie Zahlen von der Diskette geladen werden. Für den Computer endet eine Zahl, die von einem File gelesen wird, erst mit dem Ende eines Fields, was durch ein EOL-Zeichen gekennzeichnet ist. Wenn beim Lesen einer Zahl jedoch ein Komma auftritt, endet die Zahl dort, aber vom File werden noch weitere Daten gelesen, bis ein EOL-Zeichen festgestellt wird. Die Zeichen, die vom Auftreten des Kommas bis zum EOL gelesen werden, haben also keinen Einfluß mehr auf den Wert der Zahl.
Mit Hilfe des PRINT #-Befehls könne auch mehrere, durch Kommas voneinander getrennte Zahlen auf einmal gespeichert werden:

```
560 PRINT #3;1,2,3,
```

Nach der Ausführung dieses Befehls sind die gespeicherten Zahlen weder durch EOL-Zeichen noch durch Kommas getrennt. Die im PRINT #-Befehl vorkommenden Kommas sorgen lediglich dafür, daß einige zusätzliche Leerzeichen auf die Diskette geschrieben werden (siehe auch Bild 7-15).
Es ist nicht möglich, die so gespeicherten Zahlen mit Hilfe des INPUT #-Befehls eine nach der anderen zu laden und einer numerischen Variablen zuzuweisen:

```
650 INPUT #3;X
660 INPUT #3;Y
670 INPUT #3;Z
```

Genausowenig können Sie sie aber alle auf einmal laden:

```
792 INPUT #3;A,B,C
```

In beiden Fällen wird ein Fehler auftreten, weil sich zwischen den gespeicherten Zahlen keine Leerzeichen befinden dürfen.
Wenn Sie zur Trennung der Zahlen im PRINT #-Befehl statt eines Kommas ein Semikolon verwenden, werden die Zahlen direkt hintereinander, ohne Einfügung von irgendeinem anderen Zeichen, gespeichert. Beim Lesen werden deshalb nicht drei Zahlen, sondern nur eine erkannt.
Wenn nach jeder Zahl ein EOL-Zeichen gespeichert würde, gäbe es keine Probleme. Sehen Sie sich einmal das folgende Programm an:

```
300 OPEN #4,8,0,"D:ZAHLEN.DAT"
340 FOR I=1 TO 10
350 PRINT #4;I;
360 NEXT I
370 CLOSE #4
```

Die zehn Zahlen sehen, weil sie direkt hintereinander gespeichert wurden, wie eine einzige große Zahl aus. Wenn Sie allerdings das Semikolon am Ende des PRINT #-Befehls entfernen, wird nach jeder Zahl ein EOL-Zeichen gespeichert, womit der gewünschte Effekt erreicht wäre.

Wenn Sie die zu speichernden Zahlen durch Semikolons trennen, müssen Sie zusätzlich nach jeder Zahl ein Komma ausgeben. Damit dieses Komma auch als solches auf der Dis-

kette gespeichert wird, müssen Sie es in Anführungszeichen setzen. Der folgende Befehl ist ein Beispiel dafür:

```
270 PRINT #4;MENGE;",";PREIS;",";SUMME
```

Wenn Sie die Zahlen wieder laden wollen, müssen Sie darauf achten, daß das in derselben Form erfolgt wie das Speichern. In diesem Fall müßte das so aussehen:

```
450 INPUT #4;M,P,S
```

Es ist nicht möglich, die Zahlen einzeln zu laden, zum Beispiel so:

```
450 INPUT #4;M
460 INPUT #4;P
470 INPUT #4;S
```

Nachdem beim Laden der ersten Zahl ein Komma festgestellt wurde, sucht das BASIC jetzt, wie bereits beschrieben, nach einem EOL-Zeichen. Bie dieser Suche werden die zweite und die dritte Zahl gelesen, denn erst nach der dritten Zahl wurde ein EOL-Zeichen gespeichert. Folglich werden die INPUT #-Befehle in Zeile 460 und 470 nicht einwandfrei funktionieren. Sie könnten natürlich auch mit Hilfe des GET-Befehls jedes Zeichen einzeln lesen und überprüfen, ob es sich um ein Komma oder ein EOL-Zeichen handelt, aber das ist sehr viel Arbeit.
Zusammenfassend kann man sagen, daß es am besten ist, wenn Sie dafür sorgen, daß nach jeder Zahl ein EOL-Zeichen gespeichert wird. Mehr Platz wird dafür nicht benötigt und Sie ersparen sich möglicherweise eine Menge Ärger.

Datenfiles für wahlfreien Zugriff

Bei dieser Zugriffsart können Sie in einer beliebigen Reihenfolge auf jede beliebige Stelle eines Files zugreifen. Ermöglicht wird das durch die BASIC-Befehle NOTE und POINT. Diese Befehle gibt es nur bei DOS 2.0S. Wenn Sie mit DOS 1.0 arbeiten, können Sie diesen Abschnitt überspringen.

Der NOTE-Befehl

Mit Hilfe des Note-Befehls kann die momentane Position des Zeigers innerhalb eines Files festgestellt werden. Die Positionsangabe setzt sich zusammen aus der Nummer des Sektors, auf den zuletzt zugegriffen wurde und der Nummer des Zeichens innerhalb dieses Sektors, auf das zuletzt zugegriffen wurde. Die folgende Zeile ist ein Beispiel für die Benutzung des NOTE-Befehls:

```
NOTE #2,SEKT,ZEICH
```

Es wird die Zeigerposition des Files bestimmt, für das Kanal 2 geöffnet wurde. Die Nummer des Sektors, auf den zuletzt zugegriffen wurde, wird der Variablen SEKT zugewiesen. In ZEICH befindet sich nach der Ausführung des Befehls die Nummer des Zeichens innerhalb dieses Sektors, auf das zuletzt zugegriffen wurde. Die Nummer des Sektors bezieht sich nicht auf die Anzahl der Sektoren innerhalb des angesprochenen Files sondern auf die Anzahl der Sektoren auf der Diskette. Es kann hier jede Zahl von 1 bis 719 auftreten. Der erste Sektor eines Files ist also nicht unbedingt Sektor Nr. 1. Es ist außerdem auch möglich, daß die nachfolgenden Sektoren eine niedrigere Nummer haben als der erste, denn die von einem File belegten Sektoren müssen nicht unbedingt

aufeinander folgen. Der erste Sektor eines Files könnte zum Beispiel Sektor Nr. 148 sein und der zweite Nr. 153 oder Nr. 127.

Der POINT-Befehl

POINT ist das Gegenteil von NOTE. Mit Hilfe des POINT-Befehls wird der Zeiger an eine bestimmte Stelle eines Files gesetzt. Zur Festlegung dieser Stelle müssen Sie die Nummer eines Sektors und eines Zeichens innerhalb dieses Sektors angeben. Ein darauffolgender PRINT #-, PUT-, INPUT #- oder GET-Befehl wird ab dieser Stelle Daten auf die Diskette schreiben oder von der Diskette lesen. Die folgende Zeile ist ein Beispiel für die Benutzung des POINT-Befehls:

```
120 S=250:Z=10:POINT #3,S,Z
```

Der Zeiger des Files, für das Kanal 3 geöffnet wurde, wird auf das 10. Zeichen des 250. Sektors gesetzt.
Die Nummer des Sektors und des Zeichens muß mit Hilfe einer numerischen Variablen angegeben werden. Es ist nicht möglich, hier eine Konstante einzusetzen, obwohl der Wert der Variablen bei Ausführung des POINT-Befehls nicht verändert wird.
Die Nummer des Sektors muß zwischen 1 und 719 und die Zeichennummer zwischen 0 und 125 liegen (die letzten drei Bytes von jedem Sektor werden für interne Zwecke benötigt).
Bevor nicht wirklich ein Schreib- oder Lesevorgang erfolgt, wird nicht überprüft, ob der angegebene Sektor auch tatsächlich ein Teil des angegebenen Files ist. Erst wenn auf die Diskette zugegriffen wird, überprüft das Disk Operating System, ob der angegebene Sektor zu dem mit Hilfe der Kanalnummer angegebenen File gehört. Wenn das nicht der Fall ist, wird einer der beiden folgenden Fehler auftreten:

- ERROR 170, wenn mit Hilfe eines INPUT #- oder GET-Befehls gelesen werden sollte
- ERROR 171, wenn mit Hilfe eines PRINT #- oder PUT-Befehls geschrieben werden sollte

Die Benutzung des NOTE- und POINT-Befehls

Datenfiles, die für wahlfreien Zugriff ausgelegt sind, benutzen Sie mit Hilfe des NOTE- und POINT-Befehls. Wenn Sie diese Befehle an der richtigen Stelle verwenden, können Sie mit Ihrem ATARI-Computer große Datenmengen verarbeiten.
In den nächsten drei Abschnitten wird beschrieben, wie ein für wahlfreien Zugriff ausgelegtes Datenfile aufgebaut sein kann. Jede Methode hat ihre Vor- und Nachteile. Zur Benutzung solcher Files werden ziemlich lange und komplizierte Programme benötigt. Leider reicht der Platz hier nicht aus, um für jede Methode ein Beispiel zu bringen.

Indizierte Datenfiles

Bei dieser Methode werden statt eines Files zwei benutzt. In dem einen File werden in einzelnen Records die Daten gespeichert, auf die das Programm zugreifen können muß. Das zweite File übernimmt eine Indexfunktion. Für jeden Record des ersten Files ist hier ein Schlüsselwort gespeichert. Außerdem wird dort festgehalten, an welcher Stelle des ersten Files der Record beginnt. Die Angabe dieser Position erfolgt mit Hilfe einer Sektor- und einer Zeichennummer. Am Anfang Ihres Programmes werden diese Informationen zusammen mit den Schlüsselwörtern von der Diskette gelesen und Variablen (meistens Feldern) zugewiesen.
Wenn nun ein bestimmter Record gefunden werden soll, dann sucht das Programm zu-

nächst in der Indexliste das Schlüsselwort für diesen Record. Danach wird der Zeiger des ersten Files mit Hilfe der zu diesem Schlüsselwort gehörenden Sektor- und Zeichennummer und dem POINT-Befehl am Anfang des gewünschten Records positioniert. Nun kann der Record mit Hilfe des INPUT #- oder GET-Befehls gelesen oder mit Hilfe des PRINT #- oder PUT-Befehls geschrieben werden.
Wenn Sie einen Record zwischen zwei bereits bestehende Records einfügen wollen, dann ist das bei einem indizierten Datenfile eine sehr zeitaufwendige Angelegenheit. Zunächst müssen Sie, um Platz für den neuen Record zu schaffen, alle darauf folgenden Records ein Stück in Richtung Fileende bewegen. Dann tragen Sie das Schlüsselwort des neuen Records in das Indexfile ein und korrigieren die Sektor- und Zeichennummern der verschobenen Records, weil sich ja schließlich deren Position innerhalb des ersten Files beim Verschieben geändert hat. Wenn Sie einen Record löschen wollen, dann ist das genauso zeitaufwendig. Alle auf den zu löschenden Record folgenden Records müssen ein Stück in Richtung Fileanfang bewegt werden. Dann wird das Schlüsselwort des alten Records im Indexfile gelöscht und die Sektor- und Zeichennummern der verschobenen Records korrigiert.
Um Speicherplatz zu sparen, können Sie die Sektor-und Zeichennummern in einer Stringvariablen speichern. Zum Umwandeln eines Zeichens der Stringvariablen in eine Zahl und umgekehrt benutzen Sie die CHR$- und die ASC-Funktion. Sie brauchen dann nur noch zwei Zeichen, um die Sektornummer (1 bis 719) und eines, um die Zeichennummer (0 bis 125) festzuhalten.
Der große Vorteil eines indizierten Datenfiles besteht darin, daß ein bestimmter Record sehr schnell gefunden werden kann. Das liegt daran, daß ein im Speicher mit Hilfe einer Variablen festgehaltenes Schlüsselwort sehr viel schneller gefunden werden kann als wenn ein auf der Diskette gespeichertes File Record für Record durchsucht werden muß.

Verkettete Datenfiles

Die zweite Methode, ein Datenfile für wahlfreien Zugriff auszulegen, besteht darin, die gespeicherten Records miteinander zu verketten. In diesem Fall enthält jeder Record einen Zeiger, der angibt, an welcher Stelle des Files der nächste Record gespeichert ist. Dieser Zeiger besteht aus einer Sektor- und einer Zeichennummer.
Wenn Sie in einem solchen File einen bestimmten Record suchen, müssen Sie, vom ersten Record ausgehend, alle darauf folgenden Records lesen, bis Sie den gewünschten gefunden haben. Im Prinzip handelt es sich hier also um ein sequentielles File.
Der Vorteil dieses Files liegt darin, daß Sie sehr einfach neue Records hinzufügen oder alte löschen können. Wenn Sie in einem verkettetem Datenfile einen neuen Record speichern wollen, dann legen Sie ihn auf der Diskette ab und stellen mit Hilfe des NOTE-Befehls fest, an welcher Stelle des Files er gespeichert wurde. Dann müssen Sie festlegen, nach welchem Record der neue eingefügt werden soll. Diesen Record wollen wir einmal den vorhergehenden Record nennen. Der Zeiger des neuen Records muß auf die Stelle zeigen, auf die bisher der vorhergehende Record zeigte. Letzterer wird so geändert, daß er von nun an auf den neuen Record zeigt.
Wenn Sie einen Record löschen wollen, dann setzen Sie einfach den Zeiger des vorhergehenden Records auf dieselbe Stelle, auf die der Zeiger des zu löschenden Records zeigt.

Verkettete Datenfiles mit Index

In einem verketteten Datenfile mit Index können Sie einen bestimmten Record sehr schnell finden. Aber auch der Vorgang des Löschens oder des Hinzufügens eines Records läuft sehr rasch ab. Um einen bestimmten Record mit Hilfe eines Schlüsselwortes schnell zu finden, wird hier, wie bei der ersten Methode auch, ein Indexfile verwendet. Außerdem sind

die Records miteinander verkettet. Somit müssen, wenn ein Record gelöscht oder hinzugefügt wird, nur ein oder zwei Pointer geändert werden. Dieses Verfahren ist sehr kompliziert und programmintensiv. Aus diesem Grund sollten Sie sich erst dann an diese Methode der Datenspeicherung heranwagen, wenn Sie schon einige Erfahrung im Programmieren haben.

Lesen des Inhaltsverzeichnisses der Diskette

Wenn Sie das Inhaltsverzeichnis einer Diskette lesen wollen, müssen sie als zweite Zahl nach dem OPEN-Befehl eine 6 angeben. Sie können dann das Inhaltsverzeichnis der Diskette wie ein Datenfile lesen. Jedes Field dieses "Files" entspricht einer Zeile des Inhaltsverzeichnisses.
In dem OPEN-Befehl müssen Sie auch einen Filenamen angeben, der selbstverständlich auch mehrdeutig sein darf. Der Zeiger des "Files" wird nach Ausführung des OPEN-Befehls auf das erste Zeichen des ersten Fields gesetzt, das dem angegebenen Filenamen entspricht.
Lesen können Sie von diesem "File" nur die Zeilen des Inhaltsverzeichnisses, in denen ein Filenamen enthalten ist, der dem angegebenen Filenamen entspricht. Alle anderen Zeilen werden beim Lesen automatisch übersprungen. Am Ende des "Files" lesen Sie die Zeile des Inhaltsverzeichnisses, mit deren Hilfe angezeigt wird, wieviele Sektoren auf der Diskette noch frei sind, unabhängig davon, welchen Filenamen Sie angegeben haben und auch dann, wenn gar kein Filename gefunden wurde, der dem angegebenen entsprach.
Mit Hilfe des folgenden Programmes wird das ganze Inhaltsverzeichnis einer Diskette ohne Benutzung des DOS-Menues auf dem Bildschirm angezeigt:

```
10 DIM A$(20)
15 REM OPEN-Befehl zum Lesen des Inhal
tsverzeichnisses der Diskette
20 OPEN #1,6,0,"D:*.*"
30 TRAP 90:REM Wenn das Ende des Inhal
tsverzeichnisses erreicht wurde
40 INPUT #1;A$
50 PRINT A$
60 GOTO 40
90 END
```

Wenn sie nur die Filenamen lesen wollen, die einen Zusatz .BAS tragen, dann müssen Sie Zeile 20 ändern:

```
20 OPEN #1,6,0,"D:*.BAS"
```

Führen Sie keine anderen OPEN-Befehle aus, während Sie Daten aus dem Inhaltsverzeichnis einer Diskette lesen. Der Zeiger des "Files" wird sonst verschoben und die nächsten Lesevorgänge würden nicht mehr einwandfrei funktionieren.

Speicherung von Programmen, die in Maschinensprache geschrieben wurden

Mit Hilfe des DOS-Menues können auch in Maschinensprache geschriebene Programme gespeichert, geladen und ausgeführt werden. Diese Funktionen ermöglichen es, auf sogenannte Object-Files zuzugreifen, die normalerweise unter Verwendung des ATARI-Assembler/Editors erstellt werden. Es ist aber auch möglich, mit Hilfe dieser Funktionen einen

beliebigen Speicherbereich auf der Diskette abzulegen oder zu laden. Dabei spielt es absolut keine Rolle, was in diesem Bereich gespeichert ist.
Viele Anwender werden diese Funktionen nicht benutzen. Wenn Sie nicht in Assembler oder Maschinensprache programmieren, werden Sie wahrscheinlich manches, was in diesem Abschnitt beschrieben wird, nicht völlig verstehen. Wenn Sie wollen, können Sie diesen Abschnitt auch ganz überspringen.

Speichern von Binärdaten

Wenn Sie einen bestimmten Speicherbereich in binärer Form auf der Diskette ablegen wollen, wählen Sie Funktion K des DOS-Menues (BINARY SAVE). Bei DOS 1.0 und DOS 2.0S gibt es hier einige Unterschiede.

Speichern von Binärdaten unter DOS 1.0

Nachdem Sie Funktion K des DOS-Menues gewählt haben, erscheint die folgende Meldung auf dem Bildschirm:

```
SAVE-GIVE FILE,START,END
```

Nun müssen Sie einen Filenamen und die Anfangs- und Endadresse des Speicherbereiches angeben, den Sie speichern wollen. Die Angabe der Adressen erfolgt dabei in hexadezimaler Form. Hier ein Beispiel:

```
SAVE-GIVE FILE,START,END
PROGRM1.OBJ,3E00,4AFF
```

Mit Hilfe dieses Befehls wird ein Object-File mit dem Namen PROGRM1.OBJ auf der in Laufwerk 1 eingelegten Diskette gespeichert. In diesem File ist der Speicherbereich von 3E00 bis 4AFF enthalten.
Es ist auch möglich, eine Laufwerksnummer anzugeben:

```
SAVE-GIVE FILE,START,END
D2:PROGRM3.OBJ,3D70,42FF
```

Files, die mit Hilfe von Funktion K des DOS-Menues auf Diskette gespeichert wurden, werden üblicherweise unter Verwendung von Funktion L (BINARY LOAD) wieder geladen. Nach einem solchen Ladevorgang erscheint normalerweise wieder das DOS-Menue auf dem Bildschirm und Sie können eine andere Funktion wählen. Wenn Ihr Binärfile allerdings ein bestimmtes Format hat, wird es automatisch gestartet, nachdem es geladen worden ist.
Wenn Sie ein Binärfile in diesem Format speichern wollen, müssen Sie zunächst die Anfangsadresse des Programmes in den Speicherzellen 736 und 737 (2E0 und 2E1 hexadezimal) ablegen. Tun Sie das mit Hilfe des POKE-Befehls, bevor Sie das DOS-Menue aufrufen. Das niederwertige Byte der Startadresse wird in Speicherzelle 736 und das höherwertige in 737 gespeichert. Im folgenden Beispiel wird eine Anfangsadresse von 16400 (4010 hexadezimal) angenommen:

```
A=16400
READY
POKE 736,A-INT(A/256)*256
READY
POKE 737,INT(A/256)
```

Nachdem Sie die Adresse in den Speicherzellen 736 und 737 gespeichert haben, rufen Sie das DOS-Menue auf und wählen Funktion K. Bei der Eingabe des Filenamens fügen Sie die zwei Zeichen /A an. Dann wird die Anfangsadresse zusammen mit den Binärdaten auf der Diskette gespeichert:

```
SAVE-GIVE FILE,START,END
PROGRM3.OBJ/A,3CFF,4EFF
```

Speichern von Binärdaten unter DOS 2.0S

Wenn sie unter DOS 2.0S Funktion K des DOS-Menues wählen, erscheint die folgende Meldung auf dem Bildschirm:

```
SAVE--GIVE FILE,START,END(,INIT,RUN)
```

Nun müssen Sie einen Filenamen und die Anfangs- und Endadresse des Speicherbereiches angeben, den Sie speichern wollen. Die Angabe der Adressen erfolgt dabei in hexadezimaler Form. Hier ein Beispiel:

```
SAVE--GIVE FILE,START,END(,INIT,RUN)
PROGRM1.OBJ,3E00,4AFF
```

Mit Hilfe dieses Befehls wird ein Object-File mit dem Namen PROGRM1.OBJ auf der in Laufwerk 1 eingelegten Diskette gespeichert. In diesem File ist der Speicherbereich von 3E00 bis 4AFF enthalten.
Es ist auch möglich, eine Laufwerksnummer anzugeben:

```
SAVE--GIVE FILE,START,END(,INIT,RUN)
D2:PROGRM3.OBJ,3D70,42FF
```

Darüberhinaus können Sie noch zwei weitere Speicheradressen angeben. Die erste, INIT, ist die Startadresse eines Initialisierungsprogrammes. Die zweite, RUN, ist die Startadresse des darauffolgenden Hauptprogrammes. Wenn Sie ein Programm mit Hilfe von Funktion L des DOS-Menues laden, dann wird das Programm automatisch gestartet, falls Sie diese Adressen angegeben haben. Auch die Eingabe dieser Adressen erfolgt, wie Sie in dem folgenden Beispiel entnehmen können, in hexadezimaler Form:

```
SAVE--GIVE FILE,START,END(,INIT,RUN)
PROGRM3.OBJ,3CFF,4EFF,4E00,4010
```

Es ist auch möglich, nur die Startadresse des Hauptprogrammes (RUN) anzugeben:

```
SAVE--GIVE FILE,START,END(,INIT,RUN)
PROGRM4.OBJ,3E10,517F,,4800
```

Wenn Sie die Startadresse eines Initialisierungsprogrammes angeben, dann wird dieses Programm zuerst ausgeführt. Es muß mit dem Assemblerbefehl RTS enden. Danach wird an der angegebenen Startadresse mit der Ausführung des Hauptprogrammes begonnen.

Verketten von Binärfiles

Unter DOS 2.0S können Sie Binärfiles, die mit Hilfe von Funktion K des DOS-Menues (BINARY SAVE) gespeichert wurden, miteinander verketten. Rufen Sie dazu Funktion C des DOS-Menues auf (COPY FILE) und schreiben Sie hinter den Namen des Zielfiles den Zusatz /A. Das Ergebnis ist ein sogenanntes Compound-File (= zusammengesetztes File). Ein Compound-File besteht aus mehreren Binärfiles, die miteinander verkettet wurden. In einem solchen File können Sie den Inhalt von zwei oder mehr verschiedenen Speicherbereichen festhalten. Falls die Bereiche nicht unmittelbar aufeinander folgen, bleibt beim Laden des Files alles unbeeinflußt, was zwischen diesen Bereichen steht.
In einem Compound-File haben die Startadressen nicht dieselbe Funktion wie in einem normalen Binärfile. Die Funktion der Initialisierungsprogramme bleibt erhalten, aber nur das letzte Hauptprogramm wird ausgeführt. Jedesmal, nachdem ein Teil des Compound-Files geladen worden ist, wird geprüft, ob in diesem Teil die Startadresse für ein Initialisierungsprogramm vorhanden ist. Wenn das der Fall ist, dann wird dieses Programm ausgeführt, bevor der nächste Teil des Compound-Files geladen wird. Die Startadresse des Hauptprogrammes wird dem letzten Teil des Compound-Files entnommen.

Laden von Binärfiles

Mit Hilfe von Funktion L des DOS-Menues (BINARY LOAD) kann ein File geladen werden, das mit Hilfe von Funktion K oder dem Assembler/Editor gespeichert wurde. Wenn zusammen mit dem File eine Startadresse gespeichert wurde, dann erfolgt außerdem ein automatischer Programmstart.
Nachdem Sie Funktion L des DOS-Menues gewählt haben, erscheint die folgende Meldung auf dem Bildschirm:

```
LOAD FROM WHAT FILE?
```

Jetzt müssen Sie den Filenamen des Binärfiles eingeben, das Sie laden wollen. Das sieht dann zum Beispiel so aus:

```
LOAD FROM WHAT FILE?
PROGRM1.OBJ
```

Mit Hilfe dieses Befehls wird das Binärfile mit dem Filenamen PROGRM1.OBJ geladen und zwar genau an die Stelle, an der es sich auch befand, als es gespeichert wurde. Falls eine Startadresse mitgespeichert wurde, wird sofort mit der Ausführung des Programmes begonnen.
Natürlich können Sie auch angeben, von welchem Laufwerk das File geladen werden soll:

```
LOAD FROM WHAT FILE?
D2:PROGRM2.OBJ
```

Wenn Sie mit DOS 2.0S arbeiten, kann es vorkommen, daß sich der Speicherbereich, in dem sich das Menueprogramm des DOS befindet, mit dem Bereich überschneidet, in den das Binärfile geladen werden soll. In diesem Fall muß auf der in Laufwerk 1 eingelegten Diskette ein MEM.SAV-File gespeichert sein. Dann wird der Teil des Binärfiles, der sich mit dem Menueprogramm des DOS überschneidet, solange in dem MEM.SAV-File zwischengespeichert, bis das Binärfile ausgeführt wird. Wenn kein MEM.SAV-File gefunden werden kann, dann erscheint die Fehlermeldung NEED MEM.SAV TO LOAD THIS FILE (= MEM.SAV-File wird zum Laden dieses Files benötigt) auf dem Bildschirm.

Unterdrücken des automatischen Programmstarts

Wenn Sie verhindern wollen, daß ein File automatisch gestartet wird, obwohl bei der Speicherung eine Startadresse angegeben wurde, dann schreiben Sie hinter den Filenamen den Zusatz /N. Das funktioniert allerfdings nur unter DOS 2.0S. Hier ein Beispiel:

```
LOAD FROM WHAT FILE?
PROGRM3.OBJ/N
```

Ausführung eines im Speicher befindlichen Programmes

Wenn Sie ein im Speicher befindliches Maschinenprogramm starten wollen, dann wählen Sie Funktion M des DOS-Menues (RUN AT ADRESS). Danach erscheint die folgende Meldung auf dem Bildschirm:

```
RUN FROM WHAT ADDRESS?
```

Sie müssen nun die Startadresse Ihres Programmes eingeben. Die Angabe dieser Adresse erfolgt in hexadezimaler Form. Falls Sie die falsche Adresse eingeben, kann es vorkommen, daß der Computer in eine Endlosschleife gerät und die Kontrolle nicht mehr an Sie zurückgibt. In einem solchen Fall müssen Sie das System aus- und wiedereinschalten.
Der Sprung in das Maschinenprogramm erfolgt mit Hilfe des JSR-Befehls. Wenn am Ende eines solchen Programmes ein RTS-Befehl steht, dann erscheint wieder das DOS-Menue auf dem Bildschirm.

Das AUTORUN.SYS-File

Unter DOS 2.0S nimmt ein Binärfile mit dem Namen AUTORUN.SYS eine Sonderstellung ein. Wenn Sie DOS 2.0S booten und sich ein File mit diesem Namen auf der Diskette befindet, dann wird es automatisch geladen. Falls zusammen mit diesem File eine Startadresse gespeichert wurde, dann wird es auch automatisch gestartet.
Ein AUTORUN.SYS-File, das von ATARI geliefert wird, enthält ein Programm zur Abwicklung des Datenaustausches über eine RS-232-Schnittstelle. Dieses Programm benötigen Sie, wenn Sie die seriellen Ports des ATARI 850 Interface-Moduls benutzen wollen.
Es ist aber auch möglich, daß ein von Ihnen erstelltes Programm als Teil des Bootvorganges geladen und ausgeführt wird. Sie müssen dieses Programm nur mit Hilfe von Funktion K des DOS-Menues in einem Binärfile mit dem Namen AUTORUN.SYS speichern. Wenn Ihr Programm mit einem RTS endet, wird nach der Ausführung des Programmes ein systemeigenes Initialisierungsprogramm aufgerufen. Unter anderem wird darin zum Beispiel die Funktion der SYSTEM RESET-Taste festgelegt. Wenn Ihr Programm nicht mit einem RTS endet, sollte in der Speicherzelle 580 (244 hexadezimal) eine 0 und in der Speicherzelle 9 eine 1 abgelegt werden.

Modifikation von DOS 2.0S

In diesem Abschnitt werden zwei kleine Modifikationen von DOS 2.0S beschrieben. Diese Veränderungen sind nur bei DOS 2.0S möglich. Versuchen Sie deshalb nicht, DOS 1.0 auf dieselbe Art zu modifizieren.

Vergrößerung des verfügbaren Speicherplatzes

Unter DOS 2.0S können maximal vier ATARI-Diskettenstationen 810 benutzt werden. Für jedes Laufwerk wird im Speicher ein Diskettenbuffer bereitgestellt. Wenn an Ihren ATARI-Computer weniger als vier Laufwerke angeschlossen sind, können Sie den, zum Beispiel für BASIC-Programme, zur Verfügung stehenden Speicherplatz vergrößern, denn auch für jedes nicht angeschlossene Laufwerk wird ein Diskettenbuffer zur Speicherung von 128 Bytes (Zeichen) reserviert.
Diesen Speicherbereich können Sie nutzen, indem Sie die in Speicherzelle 1802 gespeicherte Zahl ändern und dem Computer so mitteilen, wieviele Laufwerke wirklich vorhanden sind.
Vergewissern Sie sich zunächst, ob sich die BASIC-ROM-Kassette im Computer befindet. Booten Sie dann das DOS, indem Sie Ihren Computer aus- und wieder einschalten. Nachdem READY auf dem Bildschirm erschienen ist, geben Sie folgendes ein:

```
?PEEK(1802)
```

Der Computer druckt daraufhin eine Codezahl aus, aus der Sie mit Hilfe von Tabelle 7-2 entnehmen können, für wieviele Laufwerke das soeben gebootete DOS 2.0S ausgelegt ist. Dann bestimmen Sie mit Hilfe der Tabelle die Codezahl für die Anzahl von Laufwerken, die wirklich an Ihren Computer angeschlossen sind. Diese Zahl legen Sie dann mit Hilfe des POKE-Befehls in Speicherzelle 1802 ab. Wenn Sie zum Beispiel den folgenden Befehl geben, ist das DOS anschließend nur noch für ein Laufwerk ausgelegt:

```
POKE 1802,1
```

Rufen Sie nun das DOS-Menue auf, indem sie den DOS-Befehl geben und legen Sie eine leere Diskette in Laufwerk 1 ein. Falls es notwendig sein sollte, formatieren Sie die Diskette unter Verwendung von Funktion I des DOS-Menues. Dann schreiben Sie das Disk Operating System mit Hilfe von Funktion H des DOS-Menues auf die Diskette. Damit haben Sie ein modifiziertes DOS erstellt. Verwenden Sie diese Diskette von nun an als neue Masterdiskette.
Wenn Sie das DOS von dieser Diskette booten, werden Sie feststellen, daß Sie mehr Speicherplatz zur Verfügung haben als wenn Sie zum Booten eine alte Diskette verwenden.

Tabelle 7-2: Anzahl von Diskettenstationen und die dazugehörige Kodezahl

Anzahl der Laufwerke	Kode-zahl*
1	1
2	3
3	7
4	15

* wird in Speicherzelle 1802 festgehalten.

Vergrößerung der Anzahl von Files, die gleichzeitig geöffnet sein dürfen

Normalerweise können nur drei Files gleichzeitig geöffnet sein. Nachdem das DOS gebootet worden ist, befindet sich in Speicherzelle 1801 die Anzahl der Files, die maximal zur selben Zeit offen sein dürfen.

Unter DOS 2.0S können Sie den Inhalt dieser Speicherzelle Ihren Anforderungen entsprechend verändern. Da für jedes offene File ein Ein/Ausgabekanal benötigt wird und nur sieben Kanäle zur Verfügung stehen, können rein gerätetechnisch nicht mehr als sieben Files gleichzeitig geöffnet sein. Allerdings müssen Sie einen Nachteil in Kauf nehmen, wenn Sie die in Speicherzelle 1801 befindliche Zahl vergrößern: für jedes zusätzliche File wird ein weiterer Filebuffer angelegt. Der dafür benötigte Speicherplatz wird beim Booten reserviert. Dadurch verringert sich der verfügbare Speicherplatz mit jedem zusätzlichen File um 128 Bytes.
Vergewissern Sie sich zunächst, ob sich die BASIC-ROM-Kassette im Computer befindet. Booten Sie dann das DOS, indem Sie Ihren Computer aus- und wieder einschalten. Nachdem READY auf dem Bildschirm erschienen ist, geben Sie folgendes ein:

```
?PEEK(1801)
```

Der Computer gibt daraufhin die Anzahl der Files aus, die maximal gleichzeitig geöffnet sein können. Normalerweise wird das eine 3 sein. Nun können Sie mit Hilfe des POKE-Befehls den Inhalt von Speicherzelle 1801 entsprechend Ihren Anforderungen verändern. Wenn Sie zum Beispiel den folgenden Befehl geben, können maximal sieben Files gleichzeitig geöffnet sein:

```
POKE 1801,7
```

Rufen Sie nun das DOS-Menue auf, indem Sie den DOS-Befehl geben und legen Sie eine leere Diskette in Laufwerk 1 ein. Falls es notwendig sein sollte, formatieren Sie die Diskette unter Verwendung von Funktion I des DOS-Menues. Dann schreiben Sie das Disk Operating System mit Hilfe von Funktion H des DOS-Menues auf die Diskette. Damit haben Sie ein modifiziertes DOS erstellt. Verwenden Sie diese Diskette von nun an als neue Masterdiskette.
Wenn Sie das DOS von dieser Diskette booten, gilt für die Anzahl der Files, die gleichzeitig offen sein können, die von Ihnen gewählte Grenze, während beim Booten von einer alten Diskette wieder die alte Grenze gilt.

Zusammenbruch der Diskettenstation

Am Ende dieses Kapitels wollen wir den schwersten Fehler beschreiben, der beim Arbeiten mit der Diskettenstation überhaupt auftreten kann, nämlich der Zusammenbruch der Diskettenstation, den sogenannten Disk Crash. Man unterscheidet dabei zwischen hard und soft Crash. Bei einem hard Crash liegt ein physikalischer Defekt der Diskette vor, also zum Beispiel ein Riß oder starke Verschmutzung. Bei einem hard Crash kann auch der Schreib/Lesekopf des Laufwerkes beschädigt werden, wodurch wieder andere Disketten Schaden nehmen würden. Aus diesem Grund sollten Sie Ihre Disketten immer sorgfältig behandeln.
Ein soft Crash tritt dann auf, wenn die auf der Diskette gespeicherten Daten verstümmelt wurden. Dieser Fehler tritt oft auf, wenn auf eine Diskette Daten geschrieben, die Diskette gewechselt und erst dann die benutzten Files geschlossen wurden. Das dabei entstehende Durcheinander läßt sich schwer beschreiben; am besten probieren Sie es einmal aus.

8

EINFÜHRUNG IN DIE GRAFIKFUNKTIONEN DES ATARI-COMPUTERS

Jedes von Ihnen erstellte Programm kann mit Hilfe der Grafikfunktionen des ATARI-Computers verbessert werden, ganz gleich, in welchem Bereich Ihr Computer eingesetzt werden soll. In diesem Kapitel werden die verschiedenen Grafikbetriebsarten (Modi), die Sie im BASIC benutzen können, beschrieben. Zusätzlich werden Sie einige Tricks zur optimalen Ausgestaltung Ihrer Programme kennenlernen.
Im BASIC gibt es zwölf verschiedene Grafikbetriebsarten, gekennzeichnet durch die Zahlen von 0 bis 11. Sie werden mit Hilfe des GRAPHICS-Befehls aktiviert, wobei Sie die Nummer der gewünschten Betriebsart hinter den Befehl stellen. Sie werden später kennenlernen, worin sich die einzelnen Betriebsarten unterscheiden. Zunächst ist es jedoch besser, wenn Sie sich auf die Grundlagen konzentrieren, auf die dann später aufgebaut werden kann.
Das, was im folgenden über den COLOR- und den SETCOLOR-Befehl gesagt wird, gilt nur für die Grafikbetriebsarten 0 bis 8. Welche Funktion diese Befehle in den Grafikbetriebsarten 9,10 und 11 haben, können Sie den entsprechenden Abschnitten dieses Kapitels entnehmen.

Die Farbregister

In jedem Grafikmodus können Sie die Farbe beziehungsweise die Farben, in der Sie auf den Bildschirm schreiben oder zeichnen (Vordergrund), die Farbe des Hintergrundes und die Farbe der Umrandung, die den Hintergrund umgibt, frei wählen. Zu diesem Zweck gibt es beim ATARI-Computer die Farbregister. Das sind Speicherzellen, in denen festgehalten wird, in welcher Farbe der Vordergrund, der Hintergrund und die Umrandung dargestellt werden. Um diesen Sachverhalt zu veranschaulichen, drücken Sie die SYSTEM RESET-Taste und geben Sie den folgenden Befehl ein:

```
SETCOLOR 2,0,0
```

Der Hintergrund ist schwarz geworden. Mit Hilfe des BASIC-Befehls SETCOLOR wurde der Wert des für die Hintergrundfarbe zuständigen Farbregisters geändert und als Folge davon erschien auf dem Bildschirm sofort die entsprechende Farbe. Nach Eingabe des folgenden Befehls erscheint die Umrandung des Hintergrundes in grün:

```
SETCOLOR 4,12,8
```

Wenn der Text schwarz sein soll, geben Sie den folgenden Befehl ein:

```
SETCOLOR 1,0,0
```

Bei den meisten anderen Personalcomputern müssen Sie, wenn Sie eine Zeichnung erstellen wollen, die Farbe, in der gezeichnet werden soll, vorher festlegen und können sie dann nicht mehr ändern. Beim ATARI-Computer hingegen können Sie mit Hilfe des SETCOLOR-Befehls jede Farbe zu jedem beliebigen Zeitpunkt verändern. So könnten Sie zum Beispiel eine unsichtbar auf dem Bildschirm erstellte Zeichnung auf einmal sichtbar machen, indem Sie nur den Wert eines Farbregisters ändern

Der SETCOLOR-Befehl

Mit Hilfe der auf den SETCOLOR-Befehl folgenden Zahlen oder arithmetischen Ausdrükken wird festgelegt, welches Farbregister welche Farbe in welcher Helligkeitsstufe darstellen soll. Die Festlegung der Farben von Vordergrund, Hintergrund und Umrandung erfolgt in jeder Grafikbetriebsart mit einem anderen Register.
Die erste Zahl nach dem SETCOLOR-Befehl bezeichnet das Register (0 bis 4). Mit Hilfe der zweiten Zahl wird die Farbe festgelegt (0 bis 15). In Tabelle 8-1 sind die Zahlen und die entsprechenden Farben nebeneinander aufgeführt. Die dritte Zahl bestimmt die Helligkeitsstufe, in der die gewählte Farbe dargestellt werden soll (0 bis 14). Hier haben nur gerade Zahlen eine Bedeutung, wobei 0 am dunkelsten und 14 am hellsten ist. Diese Zahl kann auch größer als 14 sein, aber dann wird die Differenz zu 14 nicht berücksichtigt. Wenn Sie alle Farben und Helligkeitsstufen kombinieren, erhalten Sie insgesamt 128 unterschiedliche Farbtöne. Mit Hilfe des folgenden kleinen Programms werden alle möglichen Farbschattierungen auf dem Bildschirm angezeigt:

```
10 GRAPHICS 0
20 LIST:REM Etwas auf den Bildschirm s
chreiben
30 FOR I=0 TO 15
40 SETCOLOR 2,I,0:REM Hintergrundfarbe
50 SETCOLOR 4,15-I,0:REM Farbe der Umr
andung
60 FOR J=0 TO 14 STEP 2
70 SETCOLOR 2,I,J:REM Hintergrund hell
er
80 SETCOLOR 0,0,14-J:REM Vordergrund d
unkler
90 SETCOLOR 4,15-J,J:REM Umrandung hel
ler
100 NEXT J
110 NEXT I
```

Bei jedem der fünf Farbregister gibt es für die Farbe und die Helligkeit feste Voreinstellungen. Diese Voreinstellungen sind in Tabelle 8-2 aufgeführt. Mit Hilfe des SETCOLOR-Befehls können Sie diese Werte verändern.

Tabelle 8-1: Die zur Verfügung stehenden Farben und die entsprechenden Zahlen zur Verwendung in einem SETCOLOR-Befehl.

Zahl	Farbe	Zahl	Farbe
0	Grau	8	Hellblau
1	Goldfarben	9	Blau-grün
2	Orange	10	Aqua
3	Rot	11	Grün-blau
4	Rosa	12	Grün
5	Violett	13	Gelb-grün
6	Blau-purpur	14	Orange-grün
7	Blau	15	Orange

Tabelle 8-2: Voreinstellungen der Farbregister

Register	Farbe	Helligkeit	Auf dem Bildschirm dargestellte Farbe
0	2	8	Orange
1	12	10	Aqua
2	9	4	Blau
3	4	6	Hellrot
4	0	0	Schwarz

Der COLOR-Befehl

COLOR und SETCOLOR sind leicht miteinander zu verwechseln, aber die beiden Befehle erfüllen zwei völlig unterschiedliche Funktionen. In einigen Grafikbetriebsarten gibt es mehrere Farbregister für den Vordergrund. Mit Hilfe des COLOR-Befehls wählen Sie eines dieser Farbregister aus, um in der entsprechenden Farbe zu zeichnen. In der Grafikbetriebsart 7 stehen zum Beispiel drei verschiedene Vordergrundfarben zur Verfügung. Sie bestimmen mit Hilfe des COLOR-Befehls, welche dieser drei Farben zum Zeichnen verwendet werden soll. In einigen anderen Grafikbetriebsarten gibt es allerdings nur eine Vordergrundfarbe. In diesem Fall können Sie wählen, ob Sie in dieser Farbe oder der Farbe des Hintergrundes zeichnen wollen, was allerdings dazu führen würde, daß Ihre Zeichnung unsichtbar ist.

Die auf den COLOR-Befehl folgende Zahl gibt in den meisten Grafikbetriebsarten die Nummer des gewünschten Farbregisters an. Dabei gilt immer, daß COLOR 0 das Farbregister für die Hintergrundfarbe und COLOR 1 das, beziehungsweise eines der Farbregister für die Vordergrundfarbe auswählt. In Tabelle 8-3 und 8-4 ist aufgeführt, welcher COLOR-Befehl welches Farbregister in den unterschiedlichen Grafikbetriebsarten verwendet.

In der Grafikbetriebsart 0 (normaler Textmodus) können Sie weder Linien noch Punkte zeichnen. Hier können Sie stattdessen hinter den COLOR-Befehl die Codezahl eines ATASCII-Zeichens setzen und dann mit diesem Zeichen zeichnen. Wenn Sie mit den Zeichen des normalen Zeichensatzes größere Grafiken oder Balkendiagramme erstellen wollen, dann kann diese Variante des COLOR-Befehls sehr von Nutzen sein.

Tabelle 8-3: Zusammenstellung der Grafikbetriebsarten (mit Ausnahme der Grafikbetriebsarten 9, 10 und 11, die nicht in dieses Schema passen).

Art der Grafik	Grafik-betriebs-art	Zur Verfügung stehende Farben	Bildschirm-größe (Spalten × Zeilen)	Farb-register für den Vorder-grund	Farb-regi-ster für den Hinter-grund	Farb-regi-ster für die Um-ran-dung	Auswahl der Register mit Hilfe des COLOR-Befehls	Belegter Speicher-platz (in Bytes)
Normaler Text	0	1 Farbe 2 Helligkeits-stufen	40×24	1 (Auswahl der Farbe ist nicht möglich)	2	4	*	992
Text in doppelter Breite	1	5	20×20(geteilt) 20×24(normal)	0, 1, 2, 3	4	4	(siehe Tab. 8-4)	674 (geteilt) 672 (normal)
Text in doppelter Breite und Höhe	2	5	10×20(geteilt) 12×20(normal)	0, 1, 2, 3	4	4	(siehe Tab. 8-4)	424 (geteilt) 420 (normal)
Vier-farb-grafik	3	4	40×20(geteilt) 40×24(normal)	0, 1, 2	4	4	FARBE 1: Register 0 FARBE 2: Register 1 FARBE 3: Register 2 FARBE 0: Register 4	Betriebsart 3: 434 (geteilt) 432 (norm.) Betriebsart 5: 1174(geteilt) 1176(norm.) Betriebsart 7: 4190(geteilt) 4200(norm.)
	5	4	80×40(geteilt) 80×48(normal)	0, 1, 2	4	4		
	7	4	160×80(geteilt) 160×96(normal)	0, 1, 2	4	4		
Zwei-farb-grafik	4	2	80×40(geteilt) 80×48(normal)	0	4	4	FARBE 1: Register 0 FARBE 0: Register 4	Betriebsart 4: 694 (geteilt) 696 (norm.) Betriebsart 6: 2174 (geteilt) 2184(norm.)
	6	2	160×80(geteilt) 160×96(normal)	0	4	4		
Hochauf-lösende Grafik	8	1 Farbe, 2 Hellig-keitsstufen	320×160(get.) 320×192(norm.)	1 (Auswahl der Farbe ist nicht möglich)	2	4	FARBE 1: Register 1 FARBE 0: Register 2	8112 (geteilt) 8138 (normal)

* In der Grafikbetriebsart 0 kann mit ATASCII-Zeichen gezeichnet werden. Wenn Sie zum Beispiel den Befehl COLOR ASC("!") geben, dann werden bei Ausführung eines darauf folgenden PLOT- oder DRAWTO-Befehls Ausrufungszeichen auf den Bildschirm geschrieben.

Tabelle 8-4: Ansprechen der Farbregister in den Grafikbetriebsarten 1 und 2

Zeichen	ATASCII-Kode	Angesprochenes Farbregister
Großbuchstaben, Zahlen, Sonderzeichen (! $ + –)	32-90	Normal: 0
	160-218	Invertiert*: 2
Kleinbuchstaben	61-122	Normal: 1
	225-250	Invertiert*: 3

* Auch kleine und invertiert dargestellte Buchstaben werden als Großbuchstaben angezeigt, aber trotzdem werden unterschiedliche Register angesprochen.

Der GRAPHICS-Befehl

In den Grafikbetriebsarten 1 bis 8 wird der Bildschirm bei Ausführung des GRAPHICS-Befehls in zwei Teile unterteilt. In die unteren vier Zeilen des Bildschirms können Sie normalen Text schreiben. Diesen Teil nennt man das Textfenster. Es wird nicht immer benötigt, wenn Sie eine Grafik erstellen. Sie können das Textfenster jedoch auch ganz ausblenden und haben dann mehr Platz für die Darstellung Ihrer Grafik.
Wenn Sie das Textfenster nicht benötigen, zählen Sie zu der hinter dem GRAPHICS-Befehl angegebenen Zahl 16 hinzu. Wenn Sie zum Beispiel den Befehl GRAPHICS 24 geben, dann entspricht das der Grafikbetriebsart 8 ohne Textfenster. Den Bereich, wo sich vorher das Textfenster befand, können Sie jetzt ebenfalls für Ihre Grafik verwenden, wodurch Ihnen dafür mehr Platz (in obigem Beispiel 32 Zeilen) zur Verfügung steht.
Wenn Sie zu der auf den GRAPHICS-Befehl folgenden Zahl 32 hinzuzählen, wird der Bildschirm bei der Ausführung dieses Befehls nicht, wie sonst üblich, gelöscht. Erwarten Sie aber nicht, daß der Text, den Sie in der Grafikbetriebsart 0 auf den Bildschirm geschrieben haben, bei Ausführung des Befehls GRAPHICS 40 (Grafikbetriebsart 8 + 32) in dieser Form stehen bleibt, denn in jeder Betriebsart wird der Bildschirminhalt auf eine andere Art und Weise interpretiert. Wenn Sie wissen wollen, wie Sie verschiedene Grafikbetriebsarten miteinander vermischen können, sehen Sie sich den in diesem Kapitel enthaltenen Abschnitt über die Einfügung von Text in eine Grafik an.
Wenn Sie kein Textfenster benötigen und zusätzlich der Bildschirm bei der Ausführung des GRAPHICS-Befehls nicht gelöscht werden soll, zählen Sie zu der auf den GRAPHICS-Befehl folgenden Zahl 48 hinzu. Damit kann zum Beispiel das Textfenster in eine Grafik ein- und ausgeblendet werden. Das folgende Programm ist ein Beispiel dafür:

```
10 GRAPHICS 8+16:REM GRAFIK OHNE TEXTF
ENSTER
20 COLOR 1:PLOT 0,0
30 DRAWTO 319,191:REM EINE DIAGONALE Z
EICHNEN
40 FOR ZEIT=1 TO 200:NEXT ZEIT:REM ZEI
TSCHLEIFE
50 GRAPHICS 8+32:REM TEXTFENSTER EINBL
ENDEN
60 PRINT "EINE DIAGONALE"
70 FOR ZEIT=1 TO 200:NEXT ZEIT:REM ZEI
TSCHLEIFE
80 GRAPHICS 8+48:REM TEXTFENSTER AUSBL
ENDEN
90 GOTO 40
```

Beim Umschalten der Grafikbetriebsarten flimmert der Bildschirm ein wenig, aber allzu störend dürfte das im normalen Betrieb nicht sein.

Benutzung des Textfensters

Die Eingabe oder Ausgabe von Daten mit Hilfe des INPUT- oder PRINT-Befehls läuft über das Textfenster. Falls Sie eine Grafikbetriebsart gewählt haben, bei der kein Textfenster vorhanden ist, wird der Computer den Bildschirm löschen und auf den normalen Textmodus (Grafikbetriebsart 0) umschalten, sobald ein PRINT- oder INPUT-Befehl auftritt oder eine Fehlermeldung ausgegeben werden muß. Sie können das umgehen, indem Sie mit einer Grafikbetriebsart arbeiten, in der auch Text dar-

gestellt werden kann (1 und 2). In den anderen Grafikbetriebsarten ist es hingegen um einiges schwieriger, wenn Sie eine Grafik ohne Textfenster auf dem Bildschirm darstellen und gleichzeitig in normalem Text Daten ein- oder ausgeben wollen. In solchen Fällen ist es besser, wenn Sie, wie bereits beschrieben, das Textfenster ein- und ausblenden.

Die Grafikbetriebsarten 1 und 2

Wenn Sie in einer dieser beiden Betriebsarten Text auf den Bildschirm schreiben, dann wird dieser Text gegenüber dem normalen Textmodus vergrößert dargestellt. In der Grafikbetriebsart 1 werden auf dem Bildschirm 20 Zeilen mit je 20 Zeichen dargestellt. Wenn Sie das Textfenster nicht benötigen, erhöht sich die Anzahl der Zeilen auf 24 beziehungsweise 12.

Darstellung von vergrößertem Text

In den Grafikbetriebsarten 1 und 2 werden kleine und invertiert dargestellte Buchstaben ebenfalls als Großbuchstaben auf dem Bildschirm angezeigt. Die beiden höchstwertigen Bits von jedem ATASCII-Zeichen, mit deren Hilfe normalerweise festgestellt wird, ob es sich um einen Kleinbuchstaben oder ein invertiert dargestelltes Zeichen handelt, werden hier dazu verwendet, die Farbregister auszuwählen und so festzulegen, in welcher Farbe das Zeichen dargestellt werden soll. Im Textfenster können nach wie vor alle Zeichen angezeigt werden. In diesen beiden Grafikbetriebsarten können Sie mit Hilfe des PRINT-Befehls sowohl auf den Bildschirm als auch in das Textfenster schreiben. Es muß also zwei verschiedene PRINT-Befehle geben:

```
10 GRAPHICS 1
20 PRINT #6;"VeRgRoEsSeRtEr TeXt"
30 PRINT "Normaler Text"
```

Zuerst wird der vergrößerte Text am Anfang des Bildschirms und dann der normale Text im Textfenster zur Anzeige gebracht. Jedesmal, wenn Sie vergrößerten Text auf dem Bildschirm darstellen wollen, verwenden Sie den PRINT #6-Befehl. Bei Benutzung des normalen PRINT-Befehls erfolgt die Ausgabe stattdessen im Textfenster. Wenn Sie das Textfenster nicht benötigen und stattdessen auf dem ganzen Bildschirm vergrößerten Text darstellen wollen, verwenden Sie den GRAPHICS 17–(24 Zeilen zu je 20 Zeichen) oder den GRAPHICS 18-Befehl (12 Zeilen zu je 20 Zeichen).

Funktion der Farbregister in den Grafikbetriebsarten 1 und 2

Mit Hilfe des Farbregisters 4 wird die Farbe des Hintergrundes und der Umrandung festgelegt. Es ist allerdings nicht möglich, mit einem SETCOLOR-Befehl die Farbe des dargestellten Textes zu verändern. Stattdessen ist jedes Farbregister nur für einen Teil des ATASCII-Zeichensatzes zuständig.
Wie bereits erwähnt, werden die beiden höchstwertigen Bits jedes ATASCII-Zeichens, das vergrößert dargestellt werden soll, zur Auswahl des Farbregisters verwendet. Auf den ersten Blick mag das zwar etwas seltsam erscheinen, aber dadurch können Sie den Bildschirm wesentlich dynamischer gestalten als das im normalen Textmodus (Grafikbetriebsart 0) möglich ist. Das folgende Programm wird Ihnen das auf eine sehr eindrucksvolle Art veranschaulichen.

```
10 GRAPHICS 17:REM OHNE TEXTFENSTER
20 PRINT #6;"XxXxXxXxXxXxXxXxXx"
30 PRINT #6;"x           X"
40 PRINT #6;"X  marquee  x"
50 PRINT #6;"x           X"
60 PRINT #6;"XxXxXxXxXxXxXxXxXx"
70 SETCOLOR 1,4,8:REM FARBE FUER DIE K
LEINEN X UND DEN TEXT IN DER MITTE
80 SETCOLOR 0,10,8:REM FARBE FUER DIE
GROSSEN X
90 FOR ZEIT=1 TO 150:NEXT ZEIT
100 SETCOLOR 0,4,8:REM FARBWECHSEL FUE
R DIE GROSSEN X
110 SETCOLOR 1,10,8:REM FARBWECHSEL FU
ER DIE KLEINEN X UND DEN TEXT
120 FOR ZEIT=1 TO 150:NEXT ZEIT
130 GOTO 70
```

Befehle zum Erstellen von Grafiken

Es gibt vier BASIC-Befehle (PLOT, DRAWTO, LOCATE und POSITION), mit deren Hilfe Sie in den Grafikbetriebsarten 3 bis 11 Zeichnungen erstellen können. Eigentlich können Sie diese Befehle in jeder Grafikbetriebsart benutzen, aber sie werden normalerweise nicht dazu verwendet, um Text auf dem Bildschirm auszugeben oder bestehenden Text zu verändern.

Der PLOT- und der DRAWTO-Befehl

Mit Hilfe des PLOT- und des DRAWTO-Befehls können Sie Punkte und Linien auf den Bildschirm zeichnen. Bei der Ausführung eines PLOT-Befehls wird ein einzelner Punkt auf den Bildschirm gezeichnet. Das folgenden Programm schreibt in der Grafikbetriebsart 3 unter Verwendung des Zufallsgenerators Punkte auf den Bildschirm:

```
10 GRAPHICS 3+16
20 COLOR RND(0)*4:REM FESTLEGUNG DER Z
EICHENFARBE MIT HILFE DES ZUFALLSGENER
ATORS
30 PLOT RND(0)*39,RND(0)*19:REM FESTLE
GUNG DER KOORDINATEN MIT HILFE DES ZUF
ALLSGENERATORS
40 GOTO 20
```

Die Wirkung des DRAWTO-Befehls wird durch das folgende Programm sehr gut veranschaulicht:

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,0:REM SCHWARZER HINTER
GRUND
30 COLOR 1:REM ZEICHENFARBE WAEHLEN
40 FOR Y=0 TO 191 STEP 3
50 PLOT 0,Y
60 DRAWTO Y,191
70 NEXT Y
80 GOTO 80
```

Beim DRAWTO-Befehl wird nur die Spalte und die Zeile eines Punktes angegeben, zu dem eine Linie gezeichnet werden soll. Um den Anfangspunkt dieser Linie festzulegen, steht im obigen Programm in Zeile 40 ein PLOT-Befehl mit den entsprechenden Koordinatenangaben.

Der POSITION- und LOCATE-Befehl

Mit Hilfe des POSITION-Befehls, wird, ähnlich wie beim PLOT-Befehl, ein Punkt auf dem Bildschirm adressiert, nur wird dieser Punkt bei Ausführung des POSITION-Befehls nicht gezeichnet.
Zeile 40 des obigen Programmes könnte also auch so aussehen:

```
40 POSITION 0,Y
```

Wenn Sie diese Zeile in das obige Programm einsetzen, ergibt sich allerdings ein kleiner Unterschied in der Zeichnung: die Punkte mit den Koordinaten (0,Y) werden nicht mitgezeichnet. Den POSITION-Befehl können Sie auch in den Grafikbetriebsarten, 0, 1und 2 zur Verschiebung des Cursors verwenden.
Mit Hilfe des LOCATE-Befehls können Sie feststellen, in welcher Farbe ein bestimmter Punkt des Bildschirms gezeichnet ist. Ein Beispiel dafür ist die folgende Anweisung:

```
10 LOCATE 10,14,X
```

Hiermit wird festgestellt, in welcher Farbe der Punkt an der 14. Stelle der 10. Zeile des Bildschirms gezeichnet ist. In der Variablen X wird eine Zahl abgelegt, die diese Farbe kennzeichnet. Es ist dieselbe Zahl, die vorher bei einem COLOR-Befehl angegeben wurde, um den Punkt in dieser Farbe zu zeichnen. In Tabelle 8-3 finden Sie eine Zusammenstellung dieser Zahlen.

Erstellen von Vierfarbgrafiken

In den drei Grafikbetriebsarten 3, 5 und 7 stehen Ihnen drei Farbregister für den Vordergrund und eines für den Hintergrund und die Umrandung zur Verfügung. Diese drei Betriebsarten unterscheiden sich in der Punktdichte (Auflösung) und belegen damit auch unterschiedlich viel Speicherplatz.
Das folgende Programm ist ein Beispiel dafür, daß es möglich ist, eine Zeichnung teilweise zu löschen, wenn als Zeichenfarbe die Farbe des Hintergrundes (COLOR 0) gewählt wurde:

```
10 GRAPHICS 7+16
20 SETCOLOR 2,0,0:REM SCHWARZER HINTER
GRUND
30 COLOR 1:REM ZEICHENFARBE WAEHLEN
40 SCHRITT=1:REM JEDE LINIE ZEICHNEN
50 GOSUB 100
60 COLOR 0:REM ZEICHENFARBE=FARBE DES
HINTERGRUNDES
70 SCHRITT=6:REM JEDE 6. LINIE LOESCHE
N
80 GOSUB 100
90 GOTO 90
100 REM UNTERPROGRAMM ZUM ZEICHNEN DER
 LINIEN
```

```
110 FOR Y=0 TO 95 STEP SCHRITT
120 PLOT 0,Y
130 DRAWTO Y,95
140 NEXT Y
150 RETURN
```

Zuordnung der Farbregister in den Grafikbetriebsarten 3, 5 und 7

Die Farbe des Hintergrundes und der Umrandung wird mit Hilfe von Farbregister 4 festgelegt. Für die Vordergrundfarben werden die Farbregister 0, 1 und 2 verwendet. Farbregister 3 hat keine Funktion.

Grafikbetriebsart 3

In dieser Betriebsart ist der Bildschirm in 20 Zeilen mit jeweils 40 Punkten unterteilt. Das ist die niedrigste Auflösung, mit der Sie beim ATARI-Computer Zeichnungen erstellen können. Punkte und Linien können in drei verschiedenen Farben gezeichnet werden. Mit Hilfe des COLOR-Befehls legen Sie fest, welches Farbregister zum Zeichnen benutzt werden soll. Diese Grafikbetriebsart benutzen Sie am besten für einfache Spiele oder wenn Sie große Blockbuchstaben zeichnen wollen.

Erstellen von Zweifarbgrafiken

In den beiden Grafikbetriebsarten 4 und 6 steht Ihnen ein Farbregister für den Vordergrund und eines für den Hintergrund und die Umrandung zur Verfügung. Die Auflösung ist dieselbe wie bei den Grafikbetriebsarten 5 und 7. Trotzdem wird hier weniger Speicherplatz belegt, weil bei einer Vierfarbgrafik für jeden Punkt auf dem Bildschirm zwei Bits zur Speicherung der Farbinformation benötigt werden gegenüber nur einem Bit bei einer Zweifarbgrafik. Wenn ein Bit auf 1 gesetzt ist, wird der entsprechende Punkt in der Vordergrundfarbe gezeichnet. Anderenfalls hat der Punkt dieselbe Farbe wie der Hintergrund. Aus diesem Grund wird hier fast nur halb soviel Speicherplatz benötigt wie bei den Vierfarbgrafiken.

Hochauflösende Grafik

Die höchste Auflösung erreichen Sie in der Grafikbetriebsart 8. Allerdings wird dafür auch sehr viel Speicherplatz benötigt. Wenn das Textfenster eingeblendet ist, stehen Ihnen 160 Zeilen mit jeweils 320 Punkten zur Verfügung. Ohne Textfenster erhöht sich die Zeilenzahl auf 192. In dieser Betriebsart ist die Zeichenfarbe allerdings nicht wählbar. Sie wird durch die Farbe des Hintergrundes festgelegt. Wählbar ist jedoch die Helligkeit, in der gezeichnet werden soll.

Zusätzliche Farben in der Grafikbetriebsart 8

In der Grafikbetriebsart 8 gibt es nur eine Zeichenfarbe und die unterscheidet sich noch nicht einmal von der Farbe des Hintergrundes. Es ist jedoch möglich, die dargestellten Bitmuster so zu manipulieren, daß dabei neue Farben entstehen. Bei den deutschen ATARI-Computern ist dieser Effekt wegen der deutschen Fernsehnorm leider nicht sehr ausgeprägt. Trotzdem wollen wir hier beschrei-

ben, auf welchem Prinzip die Erzeugung dieser Farben beruht und einige Beispielprogramme angeben.
In der Grafikbetriebsart 8 besteht jede Zeile aus 320 Punkten (Pixels). Jeder dieser Punkte entspricht einem halben Farbzyklus. Ein Farbzyklus entspricht der Zeit, die der Elektronenstrahl des Fernsehgerätes braucht, um zwei Pixels zu zeichnen (Bild 8-1).

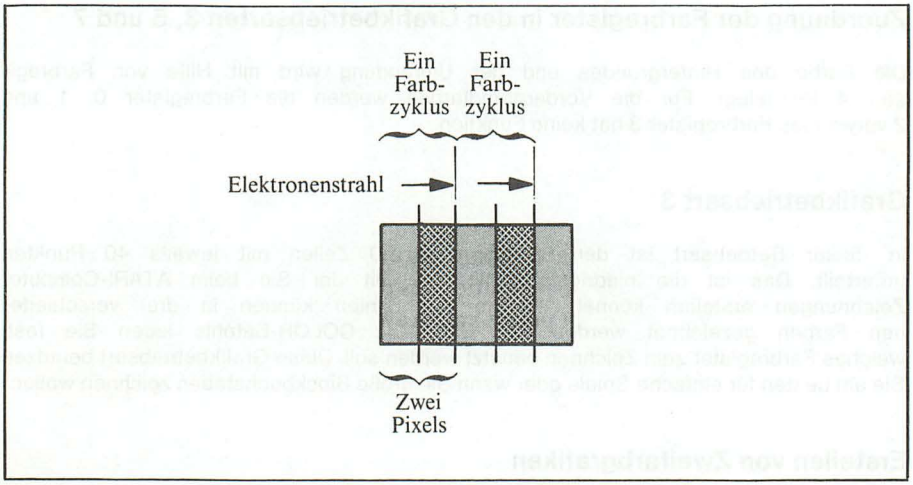


Bild 8-1: Pixels und Farbzyklen

Wenn Sie die Pixels in einer bestimmten Art und Weise aufeinander folgen lassen, ergibt sich aufgrund der Helligkeitsunterschiede beim Übergang von einem Farbzyklus auf den nächsten eine Phasenverschiebung. Diese Phasenverschiebung hat wiederum eine Farbänderung zur Folge. Der Computer selbst ändert jedoch nicht die Farbe, sondern nur die Helligkeit des Signals. In Tabelle 8-5 finden Sie eine Zusammenstellung aller Bitmuster, mit deren Hilfe Helligkeitsunterschiede und damit neue Farben erzeugt werden können. Diese Farben sind von Fernsehgerät zu Fernsehgerät unterschiedlich.
Mit Hilfe des folgenden Programmes werden die verschiedenen Bitmuster in horizontaler Richtung auf den Bildschirm geschrieben:

```
5 GRAPHICS 8:SETCOLOR 1,0,15
10 REM VIERFARB-GRAFIKBETRIEBSART 8
20 DATA 85,170,255
30 BILDSCH=PEEK(88)+PEEK(89)*256
40 READ X:PRINT "BITMUSTER=";X
80 FOR I=BILDSCH TO BILDSCH+1000
85 REM ERZEUGUNG DER VERSCHIEDENEN FAR
BEN
90 POKE I,X
100 NEXT I
105 BILDSCH=BILDSCH+1000
110 IF X=255 THEN STOP
120 GOTO 40
```

Tabelle 8-5: Erzeugung von neuen Farben mit Hilfe der durch verschiedene Bitmuster hervorgerufenen Helligkeitsunterschiede

Bitmuster	Erzeugte Farbe
00	Keine (Hintergrundfarbe)
01	Erste durch Phasenverschiebung erzeugte Farbe
10	Zweite durch Phasenverschiebung erzeugte Farbe
11	Weiß (Vordergrundfarbe)

Benutzung des PLOT- und DRAWTO-Befehls im Zusammenhang mit den zusätzlich erzeugten Farben

Geben Sie, nachdem das obige Programm gelaufen ist, einige PLOT- und DRAWTO-Befehle ein. Sie werden sehen, daß Sie auch über die zusätzlich erzeugten Farben zeichnen können. Wechseln Sie dann die Zeichenfarbe, indem Sie abwechselnd den COLOR 1- und den COLOR 0-Befehl geben. Sie können allerdings nur in der Hintergrundfarbe zeichnen. Für weitergehende Anwendungen ist diese Methode nicht verwendbar.
Sie können jedoch, während Sie in der Grafikbetriebsart 8 arbeiten, die Grafikbetriebsart 7 simulieren. In dieser Betriebsart belegt jedes Farbregister zwei Bit. Sie können dort die gewünschten Bitmuster ablegen und mit Hilfe des COLOR-Befehls dann auch in der Grafikbetriebsart 8 die Zeichenfarbe wählen und unter Verwendung des PLOT- und DRAWTO-Befehls in dieser Farbe zeichnen.
Mit Hilfe des POKE-Befehls ändern Sie die Inhalte von einigen Speicherzellen und simulieren so die Grafikbetriebsart 7, was zur Folge hat, daß Sie jetzt die Zeichenfarbe wählen können, obwohl die Bildschirmaufteilung der Grafikbetriebsart 8 entspricht. Das folgende Programm veranschaulicht diesen Sachverhalt:

```
5 DEG:REM UMSCHALTEN AUF GRAD
10 GRAPHICS 8
20 POKE 87,7:REM GRAFIKBETRIEBSART 7 S
IMULIEREN
30 SETCOLOR 2,14,14:SETCOLOR 1,0,0:REM
 FARBREGISTER DER GRAFIKBETRIEBSART 8
VERWENDEN
40 X=60:Y=40:REM KOORDINATEN FESTLEGEN
50 FOR R1=12 TO 36 STEP 3
60 COLOR 1:R=R1:GOSUB 210:REM BITMUSTE
R '01'
70 COLOR 2:R=R1+1:GOSUB 210:REM BITMUS
TER '10'
80 COLOR 3:R=R1+2:GOSUB 210:REM BITMUS
TER '11'
90 NEXT R1
100 STOP
200 REM ZEICHNEN EINES KREISES (ELLIPT
ISCHE FORM KOMMT VON DER SIMULATION DE
R GRAFIKBETRIEBSART 7)
210 PLOT X+R,Y
220 FOR WINK=0 TO 360 STEP 18
230 DRAWTO X+R*COS(WINK),Y+R*SIN(WINK)
240 NEXT WINK
250 RETURN
```

Der Nachteil dieser Methode liegt darin, daß ein Fehler auftritt, sobald Sie versuchen, mit Hilfe des PLOT- oder DRAWTO-Befehls über die in der Grafikbetriebsart 7 gültigen Bildschirmgrenzen hinauszuzeichnen. In horizontaler Richtung ist jeder Punkt jetzt zwei Pixels breit. Die Auflösung in dieser Richtung ist also nur noch halb so groß (160 Punkte). Deshalb können Sie mit Hilfe des PLOT- oder DRAWTO-Befehls auch noch über die ganze Bildschirmbreite zeichnen. In vertikaler Richtung stehen Ihnen hingegen in der Grafikbetriebsart 7 96 Zeilen und in der Grafikbetriebsart 8 192 Zeilen zur Verfügung. Obwohl die tatsächliche Bildschirmbreite halbiert wurde, bleibt die Bildschirmhöhe von 192 Zeilen erhalten. Davon können Sie die unteren 96 Zeilen aber wegen der in der Grafikbetriebsart 7 gültigen Begrenzung nicht benutzen. Es ist also doch nicht ganz so einfach, den Computer zu überlisten, aber mit einigen zusätzlichen POKE-Befehlen läßt sich das Problem lösen.

In der Speicherzelle 89 wird die Anfangsadresse des Bildschirmspeichers festgehalten. Wenn Sie den Inhalt dieser Speicherzelle verändern, können Sie auch die unteren 96 Zeilen des Bildschirms benutzen. Das folgende Programm enthält in den Zeilen 1200 bis 1290 ein Unterprogramm, mit dessen Hilfe entweder in der einen oder in der anderen Bildschirmhälfte gezeichnet werden kann.

```
10 DEG:REM UMSCHALTEN AUF GRAD
20 GRAPHICS 8
30 SETCOLOR 2,0,0
40 POKE 87,7
50 FOR M=60 TO 120 STEP 60
60 FOR I=1 TO 3
70 COLOR I
80 R=20
90 X=30+I*8+R:Y=M:REM RADIUS FESTLEGEN
100 PLT=1:GOSUB 1200
110 FOR WINK=0 TO 360 STEP 12
120 PLT=0
130 X=I*8+30+R*COS(WINK):Y=M+R*SIN(WIN
K)
140 GOSUB 1200
150 NEXT WINK
160 NEXT I
170 NEXT M
180 STOP
1190 REM ****************************
1191 REM *   UNTERPROGRAMM FUER   *
1192 REM * VIERFARBGRAFIKEN IN DER *
1193 REM *   GRAFIKBETRIEBSART 8   *
1194 REM *     Y=ZEILE (0-191)     *
1195 REM *     X=SPALTE (0-159)    *
1196 REM * PLT=0(DRAWTO),PLT=1(PLOT)*
1197 REM ****************************
1200 SA=PEEK(89)+15:REM ANFANGSADRESSE
 DES BILDSCHIRMSPEICHERS
1210 IF PLT=0 THEN GOTO 1260
1220 IF Y<96 THEN PLOT X,Y:RETURN
1230 POKE 89,SA
1240 PLOT X,Y-96
1250 GOTO 1300
1260 IF Y<96 THEN DRAWTO X,Y:RETURN
1280 POKE 89,SA
1290 DRAWTO X,Y-96
1300 POKE 89,SA-15
1310 RETURN
```

Wenn Sie dieses Unterprogramm verwenden wollen. speichern Sie die Spalte (0 bis 159) in X, die Zeile (0 bis 191) in Y und in PLT eine 1, um diesen Punkt zu zeichnen oder eine 0, um eine Linie zu diesem Punkt zu ziehen. Falls Sie letzteres tun wollen, muß vorher ein Punkt in derselben Bildschirmhälfte gezeichnet worden sein, in der sich der Punkt befindet, zu dem Sie die Linie ziehen wollen.

Die Grafikbetriebsarten 9, 10 und 11

In den Grafikbetriebsarten 9, 10 und 11 kann kein Textfenster eingeblendet werden. Aus diesem Grund können diese Grafikbetriebsarten auch nicht im Eingabemodus, sondern nur von einem Programm angesprochen werden.
Die Bildschirmaufteilung ist bei diesen drei Grafikbetriebsarten gleich groß: Es werden 192 Zeilen mit jeweils 80 Punkten dargestellt. Allein diese Tatsache ist ungewöhnlich, denn normalerweise ist die horizontale Auflösung größer als die vertikale.

Die Grafikbetriebsart 9

In der Grafikbetriebsart 9 kann mit einer Farbe in 16 verschiedenen Helligkeitsstufen auf dem Bildschirm gezeichnet werden. Zur Festlegung der Zeichenfarbe geben Sie den folgenden Befehl:

```
SETCOLOR 4,X,0
```

X ist dabei eine Zahl zwischen 0 und 15, mit deren Hilfe die Zeichenfarbe bestimmt wird (siehe Tabelle 8-1). Gleichzeitig wird mit diesem Befehl die Farbe des Hintergrundes festgelegt. Zur Auswahl der Zeichenhelligkeit dient der COLOR-Befehl, gefolgt von einer Zahl zwischen 0 und 15 (0 ist am dunkelsten und 15 am hellsten). Mit Hilfe des folgenden Programmes wird eine dunkelgrüne und eine hellgrüne Linie auf den Bildschirm gezeichnet:

```
10 GRAPHICS 9
20 SETCOLOR 4,12,0:REM FESTLEGUNG DER
ZEICHENFARBE UND DER FARBE DES HINTERG
RUNDES (GRUEN)
30 COLOR 4:REM DUNKELGRUEN
40 PLOT 39,2:DRAWTO 39,191
50 PLOT 40,2:DRAWTO 40,191
60 COLOR 12:REM HELLGRUEN
70 PLOT 41,2:DRAWTO 41,191
80 PLOT 42,2:DRAWTO 42,191
90 GOTO 90
```

Ein weiteres Anwendungsbeispiel für die Grafikbetriebsart 9 ist das folgende Programm. Hiermit wird ein Balken auf den Bildschirm gezeichnet, in dem alle 16 Helligkeitsstufen enthalten sind. Danach wird die Farbe des Balkens verändert und zwar in der Weise, daß alle 16 möglichen Farben durchlaufen werden (Zeile 100)

```
10 GRAPHICS 9
20 FOR ZEIL=64 TO 79
30 COLOR ZEIL-64
40 PLOT 2,ZEIL:DRAWTO 79,ZEIL
```

```
50 NEXT ZEIL
60 FOR ZEIL=95 TO 80 STEP -1
70 COLOR ABS(ZEIL-95)
80 PLOT 2,ZEIL:DRAWTO 79,ZEIL
90 NEXT ZEIL
100 FOR FARB=0 TO 15:SETCOLOR 4,FARB,0
:FOR ZEIT=1 TO 300:NEXT ZEIT:NEXT FARB
:GOTO 100
```

Die Grafikbetriebsart 10

Wenn im folgenden von Farben oder Zeichenfarben die Rede ist, dann ist damit immer eine Kombination von Farbe und Helligkeit gemeint. In der Grafikbetriebsart 10 kann in acht verschiedenen Farben auf dem Bildschirm gezeichnet werden. Unabhängig davon kann außerdem die Farbe des Hintergrundes bestimmt werden. Es ist allerdings nicht ganz einfach, die Zeichenfarben festzulegen. Die zur Festlegung der Zeichenfarben notwendigen Informationen werden in neun Speicherzellen (704 bis 712) gespeichert.
Daraus ergibt sich bereits, daß eine Zeichenfarbe mit Hilfe eines POKE-Befehls verändert werden kann. Dabei werden Sie aber im allgemeinen nicht genau wissen, welche Farbe in welcher Helligkeit erzeugt wird. Auf die Speicherzellen 708 bis 712 kann außerdem mit Hilfe eines SETCOLOR-Befehls zugegriffen werden.
Eine Möglichkeit, zu erfahren, welcher Wert mit Hilfe eines POKE-Befehls in einer der neun Speicherzellen abgelegt werden muß, um eine bestimmte Farbe zu erzeugen, besteht darin, die gewünschte Farbe zunächst mit Hilfe eines SETCOLOR-Befehls zu erzeugen und dann den dazugehörigen Wert mit Hilfe eines PEEK-Befehls aus der entsprechenden Speicherzelle auszulesen (Also zum Beispiel bei Verwendung von SETCOLOR 1,..,..aus Speicherzelle 709),
Die Auswahl der Zeichenfarben beim Zeichnen geschieht mit Hilfe des COLOR-Befehls. Das folgende Schema soll die soeben geschilderten Zusammenhänge noch etwas veranschaulichen:

Speicherzelle	Voreinstellung	Beeinflussung mit Hilfe des POKE-Befehls oder	Auswahl mit Hilfe von
704	0	0	COLOR 0 (Hintergrundfarbe)
705	0	–	COLOR 1
706	0	–	COLOR 2
707	0	–	COLOR 3
708	40	SETCOLOR 0,..,..	COLOR 4
709	202	SETCOLOR 1,..,..	COLOR 5
710	148	SETCOLOR 2,..,..	COLOR 6
711	70	SETCOLOR 3,..,..	COLOR 7
712	0	SETCOLOR 4,..,..	COLOR 8

Mit Hilfe des folgenden Programmes werden verschiedenfarbige Balken auf den Bildschirm gezeichnet und anschließend die Farbe der Balken verändert:

```
10 GRAPHICS 10
20 FOR SPAL=15 TO 65
30 X=INT(RND(0)*8)+1
40 COLOR X:REM FESTLEGUNG DER ZEICHENF
ARBE DES BALKENS (ZUFALLSBEDINGT)
50 PLOT SPAL,2:DRAWTO SPAL,191
```

```
60 NEXT SPAL
70 REM MIT HILFE DES FOLGENDEN PROGRAM
MTEILES WERDEN DIE FARBEN DER BALKEN Z
UFALLSBEDINGT VERAENDERT
80 FOR FARBE=704 TO 712
90 X=INT(RND(0)*255)
100 POKE FARBE,X
110 NEXT FARBE
120 FOR ZEIT=1 TO 100:NEXT ZEIT
130 GOTO 80
```

Die Grafikbetriebsart 11

In der Grafikbetriebsart 11 kann in 16 verschiedenen Farben auf dem Bildschirm gezeichnet werden. Die Helligkeit der Farben kann allerdings nicht unabhängig voneinander, sondern nur gemeinsam beeinflußt werden. Die Zeichenfarbe wird mit Hilfe eines COLOR-Befehls, gefolgt von einer Zahl von 0 bis 15, bestimmt. Die damit ausgewählten Farben sind dieselben, die in Tabelle 8-1 aufgeführt sind. Die Helligkeit der auf dem Bildschirm dargestellten Farben kann mit Hilfe des folgenden Befehls beeinflußt werden.

```
SETCOLOR 4,0,X
```

Dabei muß für X eine ganze Zahl zwischen 0 und 14 eingesetzt werden (0 ist am dunkelsten). Mit Hilfe des folgenden Programmes wird zunächst eine Art Regenbogen gezeichnet und dann die Helligkeit dieses Regenbogens verändert (Zeile 60):

```
10 GRAPHICS 11
20 FOR ZEIL=0 TO 15
30 COLOR ZEIL
40 PLOT 0,ZEIL:DRAWTO 79,ZEIL
50 NEXT ZEIL
60 FOR HELL=0 TO 14:SETCOLOR 4,0,HELL:
FOR ZEIT=1 TO 35:NEXT ZEIT:NEXT HELL:G
OTO 60
```

Und hier noch ein anderes Programm, mit dessen Hilfe ein sehr interessanter Effekt erzeugt werden kann:

```
10 GRAPHICS 11
20 FOR ZEIL=0 TO 191
30 COLOR ZEIL/12
40 PLOT 0,ZEIL
50 DRAWTO 79,ZEIL
60 NEXT ZEIL
70 FOR HELL=0 TO 14 STEP 2:SETCOLOR 4,
0,HELL:FOR ZEIT=1 TO 50:NEXT ZEIT:NEXT
 HELL
80 FOR HELL=14 TO 0 STEP -2:SETCOLOR 4
,0,HELL:FOR ZEIT=1 TO 50:NEXT ZEIT:NEX
T HELL:GOTO 70
```

Einfügen von Text in eine Zeichnung

Es ist zwar möglich, mit Hilfe des Textfensters gleichzeitig einen Text und

eine Zeichnung auf den Bildschirm zu bringen, aber es gibt keine Funktion, die es Ihnen ermöglicht, einen Text direkt in eine Zeichnung einzufügen. Bei Zweifarbgrafiken (Grafikbetriebsart 4, 6 oder 8) gibt es allerdings einen recht einfachen Weg, wie Sie das trotzdem tun können. Es wird dabei der im Speicher abgelegte Zeichensatz benutzt.
Wenn Sie die in der Speicherzelle 756 gespeicherte Zahl mit 256 multiplizieren, erhalten Sie die Anfangsadresse des Zeichensatzes. Jedes Zeichen belegt acht Bytes. Die Aufgabe besteht nun darin, das gewünschte Zeichen zu lokalisieren und danach, Byte für Byte, an die gewünschte Stelle in der Zeichnung zu übertragen. Das folgenden Programm erledigt diese Aufgabe:

```
10 DIM TXT$(64)
20 GRAPHICS 8
30 INPUT X,Y,TXT$
50 GOSUB 2000
60 GOTO 30
1995 REM UNTERPROGRAMM ZUM EINFUEGEN V
ON TEXT IN EINE ZEICHNUNG
2000 SA=PEEK(89)*256+PEEK(88):REM ANFA
NG DES BILDSCHIRMSPEICHERS
2010 MOD=PEEK(87):REM FESTSTELLEN DER
GRAFIKBETRIEBSART
2020 IF MOD=8 THEN SPAL=40:ZEIL=192
2030 IF MOD=6 THEN SPAL=20:ZEIL=96
2040 IF MOD=4 THEN SPAL=10:ZEIL=24
2050 IF Y>ZEIL OR X>SPAL THEN RETURN
2060 ANF=SA+Y*SPAL+X:REM ANFANGSADRESS
E FUER DEN TEXT
2070 FOR E1=1 TO LEN(TXT$)
2080 GOSUB 2200
2090 ZEICH=PEEK(756)*256:REM ANFANGSAD
RESSE DES ZEICHENSATZES
2100 ZEICH=ZEICH+E3*8
2110 FOR E2=7 TO 0 STEP -1
2120 POKE ANF+E2*SPAL,PEEK(ZEICH+E2)
2130 NEXT E2
2140 X=X+1:IF X>=SPAL THEN ANF=ANF+SPA
L*8:X=0:REM NAECHSTE ZEILE
2150 ANF=ANF+1
2160 NEXT E1
2170 RETURN
2195 REM UMWANDLUNG DER ATASCII-ZEICHE
N
2200 E3=ASC(TXT$(E1,E1))
2210 IF (E3<32) OR (E3>127 AND E3<160)
 THEN E3=E3+64:RETURN
2220 IF E3>31 AND E3<96 THEN E3=E3-32
2230 RETURN
```

Mit Hilfe der Variablen X und Y wird festgelegt, ab wo die Textausgabe auf dem Bildschirm erfolgen soll. In X wird die Spalte und in Y die Zeile des entsprechenden Punktes gespeichert. TXT$ kann selbstverständlich auch für eine andere Länge dimensioniert werden.
In den Zeilen 2000 bis 2060 wird festgestellt, welche Grafikbetriebsart gewählt wurde und damit die Anzahl der Zeichen pro Zeile bestimmt. Außerdem wird mit Hilfe der angegebenen Koordinaten die Bildschirmadresse berechnet, ab der der Text gespeichert werden muß. Das Unterprogramm in den Zeilen 2200 bis 2230 berechnet

aus den Kodezahlen der in TXT$ abgelegten Zeichen einen Offset. Dieser Offset entspricht dem Abstand von der Anfangsadresse des Zeichensatzes bis zu dem gewünschten Zeichen (in Bytes).
Die Zeilen 2100 bis 2130 übertragen die acht Byte jedes Zeichens vom Zeichensatz an die entsprechende Stelle auf dem Bildschirm. Wenn eine Zeile vollgeschrieben ist, erfolgt ein "Zeilenvorschub" (Zeile 2140).
In TXT$ ist der Text gespeichert, der in eine Zeichnung eingeblendet werden soll und X und Y geben die Stelle an, an der das erste Zeichen dieses Textes angezeigt werden soll. In der Grafikbetriebsart 4 können pro Zeile 10, in der Grafikbetriebsart 6 20 (wie in der Grafikbetriebsart 2) und in der Grafikbetriebsart 8 40 Zeichen dargestellt werden. In diesem Bereich muß sich auch der Wert von X bewegen. Die Zeilennummer (Y) muß zwischen 0 und der in der entsprechenden Grafikbetriebsart gültigen maximalen Zeilenzahl liegen. In vertikaler Richtung ist dieses Programm also sehr flexibel ausgelegt.
Wegen der PEEK- und POKE-Befehle ist dieses Programm nicht gerade besonders schnell. Sie können den Datentransfer vom Zeichensatz zum Bildschirm aber zum Beispiel schon dadurch beschleunigen, indem Sie für die Berechnung des Offset ein kleines Assemblerprogramm schreiben.
Mit diesem Programm können Sie Groß- und Kleinbuchstaben sowie Grafikzeichen auf den Bildschirm bringen. Für invertiert dargestellte Zeichen ist dieses Programm nicht geeignet.

Füllen bestimmter Bildschirmteile mit einer Farbe

Zusätzlich zu den bisher besprochenen BASIC-Befehlen zum Erstellen von Zeichnungen gibt es noch einen anderen Befehl, der direkt das Operating System anspricht, den XIO-Befehl. Mit Hilfe dieses Befehls können bestimmte Bildschirmanteile mit einer Farbe gefüllt werden. Allerdings bedarf es vor der Anwendung dieses Befehls einer gewissen Vorbereitung. Im folgenden Beispiel wird das mit Hilfe einiger BASIC-Befehle gemacht:

```
10 GRAPHICS 7
20 COLOR 1
30 PLOT 70,40
40 SETCOLOR 2,0,0
50 DRAWTO 35,0
60 DRAWTO 34,0
70 POSITION 0,40
80 POKE 765,1
90 XIO 18,#6,0,0,"S:"
```

Die Ausführung des eigentlichen XIO-Befehls geschieht erst in den Zeilen 80 und 90. Der POKE-Befehl in Zeile 80 hat dieselbe Funktion wie sonst der COLOR-Befehl. Die dort angegebenen Zahlen entsprechen denen, die nach einem COLOR-Befehl angegeben werden und können somit ebenfalls Tabelle 8-3 entnommen werden. Die Farbe, mit der ein Bildschirmteil gefüllt wird, kann, wie beim Zeichnen von Punkten oder Linien, mit Hilfe des SETCOLOR-Befehls verändert werden. Der XIO-Befehl selbst hat immer dasselbe Format wie in Zeile 90 des obigen Programmes. Sie können ihn direkt übernehmen, wenn Sie ein eigenes Programm schreiben.

Anwendung des XIO-Befehls

Der XIO-Befehl ist zum Füllen einer viereckigen Fläche vorgesehen. Das Ergebnis des obigen Programmes werden Sie jedoch wahrscheinlich zunächst für ein

Dreieck halten. Aber wenn Sie sich das Programmlisting einmal etwas genauer ansehen, werden Sie feststellen, daß es sich bei dem vermeintlichen Dreieck in Wirklichkeit um ein Viereck mit einer gegenüber den drei anderen Seiten sehr kleinen vierten Seite handelt (Zeile 50–60). Das Ergebnis eines XIO-Befehls ist nur dann vorsehbar, wenn Sie sich genau an die folgenden Anweisungen halten:

1. Zeichnen Sie mit Hilfe des PLOT-Befehls einen Punkt an der unteren rechten Ecke des zu füllenden Bildschirmteiles.
2. Zeichnen Sie eine Linie zur oberen rechten Ecke.
3. Zeichnen Sie eine Linie zur oberen linken Ecke.
4. Gehen Sie mit Hilfe eines POSITION-Befehls an die untere linke Ecke.
5. Bestimmen Sie die Farbe, mit der die Figur gefüllt werden soll. Legen Sie dazu in der Speicherzelle 765 mit Hilfe des POKE-Befehls diesselbe Zahl ab, die Sie sonst bei einem COLOR-Befehl angeben würden.
6. Geben Sie nun den XIO-Befehl in der folgenden Form: XIO 18, #6,0,0,"S:"

Die ersten vier Anweisungen können auch in umgekehrter Reihenfolge gegeben werden. Sonst dürfen jedoch keine Anweisungen vertauscht werden. Das Ergebnis des XIO-Befehls wäre sonst nicht vorhersehbar. Es gelten aber auch noch andere Einschränkungen. Wenn zum Beispiel innerhalb des zu füllenden Bildschirmteiles ein Punkt gezeichnet wurde, dann wird die Ausführung des XIO-Befehls bei Erreichen dieses Punktes abgebrochen. Zur Veranschaulichung dieser Tatsache fügen Sie einmal die folgende Zeile in das obige Programm ein:

```
15 PLOT 25,25
```

Wenn die zu füllende Figur an der unteren rechten Ecke begonnen wurde, wird sie von oben her gefüllt. Sonst erfolgt der Füllvorgang von unten her. Der XIO-Befehl ist zwar ziemlich schnell, aber auch ziemlich aufwendig. Trotzdem können mit seiner Hilfe Zeichnungen sehr viel schneller erstellt werden.

Anwendung der Grafikbetriebsarten

Die BASIC-Programme in Bild 8-2, 8-3 und 8-4 sind Beispiele für die Anwendung der Grafikbetriebsarten. In Bild 8-2 sehen Sie, daß auch die Grafikbetriebsart 0 zur Erstellung von Grafiken benutzt werden kann. Das Programm in Bild 8-3 ist für die Dateneingabe in den Grafikbetriebsarten 1 oder 2 gedacht. Vergleichen Sie dieses Programm einmal mit dem in Kapitel 4 vorgestellten Unterprogramm zur Eingabe von Strings in der Grafikbetriebsart 0 (Bild 4-37). Das in Bild 8-4 vorgestellte Programm zur Berechnung der Ausgleichsgeraden war ursprünglich für einen anderen Computer geschrieben. Es wurde jedoch durch den Zusatz von einigen Grafikbefehlen für den ATARI-Computer optimiert. Es werden jetzt nicht nur nackte Zahlen, sondern auch eine sehr schnelle anschauliche Zweifarbgrafik als Ergebnis ausgegeben.
Wenn Sie etwas mehr Erfahrung in der Programmierung von Grafiken haben, werden Sie noch bessere Anwendungen für die Grafikbetriebsarten finden. In Kapitel 9 werden Sie andere Methoden der Programmierung von Grafiken finden, die Ihnen nur der ATARI-Computer bietet.

```
10 REM PROGRAMM ZUm ERSTELLEN EINES
11 REM      BALKENDIAGRAMMES
12 REM
13 REM
15 REM ANZUZEIGENDE DATEN: MONAT UND
16 REM DER UMSATZ DIESES MONATS
17 REM
20 DATA JAN,800,FEB,820,MAE
21 DATA 765,APR,779,MAI,610
22 DATA JUN,650,JUL,780,AUG
23 DATA 800,SEP,825,OKT,840
24 DATA NOV,870,DEZ,910
25 REM
26 REM
30 GRAPHICS 0:SETCOLOR 2,12,12
35 SETCOLOR 1,0,0
40 P=12:REM ANZAHL DER MONATE
50 ZEILEN=20:SPALTEN=30
55 REM
75 REM
80 DIM MONAT$(P*3),X$(3),UMSATZ(P)
87 REM
88 REM LESEN DER DATEN
89 REM
90 FOR I=1 TO P
100 READ X$:REM NAME DES MONATS LESEN
110 MONAT$(LEN(MONAT$)+1)=X$
120 READ ZAHL
130 UMSATZ(I)=ZAHL
137 REM
138 REM HOECHSTEN UND NIEDRIGSTEN
139 REM UMSATZ BESTIMMEN
140 IF UMSATZ(I)>=MAX THEN MAX=UMSATZ(
I)
150 IF UMSATZ(I)<=MIN OR MIN=0 THEN MI
N=UMSATZ(I)
160 NEXT I
167 REM
168 REM ZEICHENMASSSTAB FESTLEGEN
170 MITTE=(MAX+MIN)/2:REM ERWARTETER M
ITTELWERT
180 MASS=INT(MAX-MIN)/ZEILEN
190 ZRAUM=INT(SPALTEN/P):REM ZWISCHENR
AUM
191 REM
200 POKE 752,1:PRINT "UMSATZ VON 1983"
211 REM ERSTELLEN DER ZEICHNUNG
220 FOR I=1 TO P
230 GOSUB 1000
240 NEXT I
250 FOR I=1 TO P
260 POSITION 31,I+3
270 M=(I-1)*3+1
280 PRINT MONAT$(M,M+2);" ";UMSATZ(I);
```

Bild 8-2: Erstellen eines Balkendiagrammes

```
300 NEXT I
310 GOTO 310
1000 REM UNTERPROGRAMM ZUM ZEICHNEN
1010 X=I*ZRAUM
1020 Y=((MAX-(MITTE+UMSATZ(I))/2))/MAS
S
1025 COLOR ASC(" ");REM ZEICHEN, MIT D
EM GEZEICHNET WERDEN SOLL
1030 PLOT X,Y
1040 DRAWTO X,ZEILEN
1050 FOR J=1 TO 3
1060 POSITION X,ZEILEN+J
1070 M=(I-1)*3+J
1080 PRINT MONAT$(M,M);
1090 NEXT J
1100 RETURN
```

Bild 8-2: Erstellen eines Balkendiagrammes (Fortsetzung)

```
10 LAENGE=7:ZEILE=9:SPALTE=10
15 GRAPHICS 1
20 POSITION 0,ZEILE:PRINT #6;"EINGABE"
30 GOSUB 1200
40 STOP
1200 REM ****************************
1210 REM *  DATENEINGABE IN DER   *
1220 REM *GRAFIKBETRIEBSART 1 ODER 2*
1230 REM *--------------------------*
1240 REM *LAENGE=MAX. EINGABELAENGE *
1250 REM *ZEILE=EINGABEZEILE      *
1260 REM *SPALTE=EINGABESPALTE    *
1270 REM ****************************
1280 DIM D$(LAENGE)
1290 IF TASTOPEN=0 THEN OPEN #1,4,0,"K
:":TASTOPEN=1
1300 POSITION SPALTE,ZEILE
1310 FOR I=1 TO LAENGE
1320 ? #6;"_";
1330 NEXT I
1340 POSITION SPALTE,ZEILE
1350 REM LESEN DER EINGABEDATEN VON DE
R TASTATUR
1360 GET #1,X
1370 IF X>=32 AND X<=95 AND LEN(D$)<LA
ENGE THEN D$(LEN(D$)+1)=CHR$(X):? #6;C
HR$(X);:GOTO 1360
1380 IF X<>126 THEN 1460:REM UEBERSPRI
NGEN, FALLS KEIN BACKSPACE EINGEGEBEN
WURDE
```

Bild 8-3: Dateneingabe in der Grafikbetriebsart 1 oder 2

```
1390 IF LEN(D$)<1 THEN 1460:REM BACKSP
ACE NICHT DURCHFUEHREN, WENN KEINE EIN
GABEDATEN VORHANDEN SIND
1400 IF LEN(D$)=1 THEN D$=""
1410 IF LEN(D$)>1 THEN D$=D$(1,LEN(D$)
-1)
1420 POSITION SPALTE+LEN(D$),ZEILE
1430 PRINT #6;"_";
1440 POSITION SPALTE+LEN(D$),ZEILE
1460 IF X=155 THEN RETURN
1470 GOTO 1360
```

Bild 8-3: Dateneingabe in der Grafikbetriebsart 1 oder 2 (Fortsetzung)

```
0 REM ERMITTELN UND ZEICHNEN DER
1 REM    AUSGLEICHSGERADEN
2 REM
3 REM  ZUR ANPASSUNG AUF DEN ATARI
4 REM WURDEN FOLGENDE UNTERPROGRAMME
5 REM        EINGEFUEGT:
6 REM AB 800: SETZEN DER PARAMETER
7 REM       ZUM ZEICHNEN
8 REM AB 900: ERMITTELN DES MASSSTABS
9 REM AB 1000: ZEICHNEN DER KURVE
10 GRAPHICS 0
15 PRINT "ERMITTELN DER AUSGLEICHSGERA
DE"
20 PRINT
30 PRINT "ANZAHL DER PUNKTE";
40 INPUT N:DIM XY(N,2)
50 GOSUB 700:REM INITIALISIERUNG
60 GOSUB 800:REM SETZEN DER PARAMETER
99 REM EINGABE DER PUNKTKOORDINATEN
100 FOR I=1 TO N
110 PRINT "X- UND Y-WERT VON PUNKT ";I
;
120 INPUT X,Y:XY(I,1)=X:XY(I,2)=Y
125 GOSUB 850:REM ERMITTELN DER MAXIMA
LWERTE
129 REM ZWISCHENSUMMEN BERECHNEN
130 J=J+X
140 K=K+Y
150 L=L+X^2
160 M=M+Y^2
170 R2=R2+X*Y
180 NEXT I
189 REM BERECHNUNG DER KOEFFIZIENTEN
190 B=(N*R2-K*J)/(N*L-J^2)
200 A=(K-B*J)/N
220 PRINT "F(X) = ";A; + (";B;" * X)"
```

Bild 8-4: Ermitteln und Zeichnen der Ausgleichsgeraden

```
229 REM ANALYSE DER ABWEICHUNG
230 J=B*(R2-J*K/N)
235 GOSUB 900
240 M=M-K^2/N
250 K=M-J
260 PRINT
270 R2=J/M
280 PRINT "DETERMINATIONSKOEFFIZIENT (
R^2):"
282 PRINT R2
283 PRINT
290 PRINT "KORRELATIONSKOEFFIZIENT:"
291 PRINT SQR(R2)
292 PRINT
300 PRINT "STANDARDABWEICHUNG:"
301 PRINT SQR(K/(N-2))
310 PRINT
340 PRINT "ZUM ERSTELLEN DER ZEICHNUNG
 DRUECKEN"
345 PRINT "SIE BITTE IRGENDEINE TASTE.
";
350 OPEN #1,4,0,"K:"
360 GET #1,X9
370 GOSUB 1000
380 GOTO 380
699 REM FELDVARIABLE INITIALISIEREN
700 FOR I=1 TO N:XY(I,1)=0:XY(I,2)=0:N
EXT I
710 YMAX=0
720 XMAX=0
750 RETURN
799 REM SETZEN DER PARAMETER ZUM ZEICH
NEN
800 ZEILEN=79
810 SPALTEN=159
819 REM GRAFIKBETRIEBSART 7: 80 ZEILEN
 ZU JE 160 PUNKTEN
820 RETURN
849 REM MAXIMALWERTE VON X UND Y BEREC
HNEN
850 IF XY(I,1)>XMAX THEN XMAX=XY(I,1)
860 IF XY(I,2)>YMAX THEN YMAX=XY(I,2)
890 RETURN
899 REM BERECHNEN DES ZEICHENMASSSTABS
900 YMASS=YMAX/ZEILEN
910 XMASS=XMAX/SPALTEN:REM ZWISCHENRAU
M
920 RETURN
999 REM UNTERPROGRAMM ZUM ZEICHNEN
1000 GRAPHICS 7
1010 SETCOLOR 0,12,2:REM FARBE VON X-
UND Y-ACHSE
1019 REM X- UND Y-ACHSE ZEICHNEN
1020 PLOT 0,0:DRAWTO 0,ZEILEN
```

Bild 8-4: Ermitteln und Zeichnen der Ausgleichsgeraden (Fortsetzung)

```
1030 DRAWTO SPALTEN,ZEILEN
1040 COLOR 2:REM FUER DIE PUNKTE
1050 FOR I=2 TO N
1060 PLOT INT(XY(I,1)/XMASS),ZEILEN-IN
T(XY(I,2)/YMASS):NEXT I
1069 REM ZEICHNEN DES TRENDS
1070 COLOR 3
1080 PLOT 0,ZEILEN-(A/YMASS):REM SCHNI
TTPUNKT MIT DER ACHSE
1090 FOR I=1 TO N
1100 Y=A+XY(I,1)*B:DRAWTO XY(I,1)/XMAS
S,ZEILEN-(Y/YMASS)
1110 NEXT I
1120 POSITION 0,0:PRINT "VORGEGEBENE D
ATEN=GELB; TREND=BLAU"
1130 PRINT "FUNKTIONSGLEICHUNG:":PRINT
 "F(X)=";B;"X + ";A
1140 RETURN
```

Bild 8-4: Ermitteln und Zeichnen der Ausgleichsgeraden (Fortsetzung)

9

WEITERFÜHRENDE BESCHREIBUNG DER GRAFIKFUNKTIONEN DES ATARI-COMPUTERS

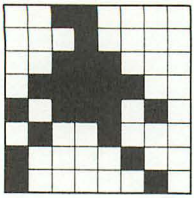
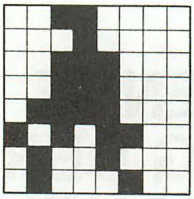
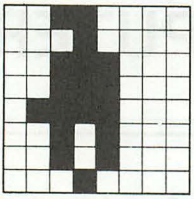
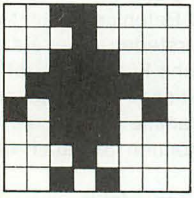
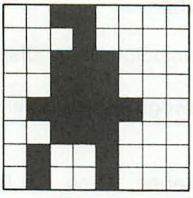
Im vorigen Kapitel wurden die Grafikfunktionen des ATARI-Computers behandelt, die zusammen mit dem BASIC verwendet werden. Die Grafikfunktionen, die in diesem Kapitel beschrieben werden, sind hingegen gewissermaßen Bestandteile der Hardware des Computers und können deshalb nicht direkt mit Hilfe von BASIC-Befehlen benutzt werden. Aus diesem Grund wird dieses Kapitel auch schwerer zu verstehen sein als das vorhergehende. Der ATARI-Computer bietet Ihnen eine unglaubliche Fülle von Grafikfunktionen, aber es wird sicher nicht einfach für Sie sein, all diese Funktionen zu verstehen und zu benutzen. In diesem Kapitel werden Sie viele Beispielprogramme finden, die Ihnen diese komplizierte Materie näher bringen sollen. Einige davon können Sie auch als Unterprogramme in selbst erstellte Programme einbauen.
Im Folgenden sind die Hauptthemen dieses Kapitels zusammengefaßt:

- Verändern des Zeichensatzes zur Erzeugung von Bewegung
- Realisierung eines beliebigen Bildschirmformats durch Manipulation der Display List
- Benutzung von Player-Missile-Grafiken zur schnellen Bewegung von Objekten auf dem Bildschirm

Das ist nur ein Teil der Möglichkeiten, die Ihnen offen stehen, wenn Sie Ihren ATARI-Computer erst einmal genau kennengelernt haben.

Verändern des Zeichensatzes

Im Zeichensatz werden alle Zeichen, die vom Computer dargestellt werden können, in

	Bitmuster	Dezimal
	00110000	48
	00010000	16
	00111000	56
	01111100	124
	10111010	186
	01001000	72
	10000100	132
	10000010	130
	00110000	48
	00010000	16
	00111000	56
	00111000	56
	01111000	120
	10101100	172
	01001000	72
	01000100	68
	00110000	48
	00010000	16
	00111000	56
	00111000	56
	01111000	120
	00101000	40
	00101000	40
	00010000	16
	00110000	48
	00010000	16
	00111000	56
	01111100	124
	10111010	186
	00111000	56
	00010000	16
	00101000	40
	00110000	48
	00010000	16
	00111000	56
	00111000	56
	01111100	124
	00111000	56
	01001000	72
	01001000	72

Anmerkung: Jedes Zeichen belegt in vertikaler Richtung acht Bildschirmzeilen. Pro Zeichen und Bildschirmzeile wird ein Byte gespeichert, mit dessen Hilfe festgelegt wird, welche acht Punkte innerhalb dieser Zeile hell und welche dunkel bleiben, damit das entsprechende Zeichen dargestellt wird.

Bild 9-1: Zeichen und die dazugehörigen Bitmuster

binärer Form gespeichert. Der Standardzeichensatz befindet sich im ROM und beginnt mit der Speicherzelle 57344 (E000 hexadezimal). In Speicherzelle 756 (2F6 hexadezimal) ist die Anfangsadresse des Zeichensatzes gespeichert. Diese Speicherzelle wird deshalb auch das Character Address Base Register, abgekürzt CHBAS, genannt. Normalerweise wird mit Hilfe von CHBAS die Speicherzelle 57344 addressiert. Es ist jedoch auch möglich, in CHBAS die Anfangsadresse eines Zeichensatzes festzuhalten, den Sie selbst erstellt und im Speicher abgelegt haben. Damit können Sie den Standardzeichen ein anderes Aussehen geben oder eigene Zeichen erfinden und damit Grafiken erstellen. Ein Beispiel dafür sind die in Bild 9-1 aufgeführten "Zeichen". Mit Hilfe dieser fünf Zeichen kann eine einfache, in fünf Stufen bewegte Figur auf den Bildschirm gebracht werden.
Für jede dieser fünf Stufen bauen Sie nun ein Zeichen auf und speichern die zur Darstellung dieses Zeichens notwendige Information mit Hilfe des POKE-Befehls ab. Dann verändern Sie den Inhalt von CHBAS so, daß an Stelle des Standardzeichensatzes die neuen Zeichen adressiert werden. Das ist alles, was Sie tun müssen, um die gewünschte Bewegung auf dem Bildschirm ablaufen zu lassen. Dem folgenden Programm können Sie entnehmen, wie das im einzelnen gemacht wird:

```
1 REM ERZEUGUNG VON BEWEGUNG DURCH
2 REM VERAENDERN DES ZEICHENSATZES
5 DIM ZSATZ(5)
10 DATA 48,16,56,124,186,72,132,130
20 DATA 48,16,56,56,120,172,72,68
30 DATA 48,16,56,56,120,40,40,16
40 DATA 48,16,56,124,186,56,16,40
50 DATA 48,16,56,56,124,56,72,72
51 REM DAMIT IST DER ZEICHENSATZ DEFIN
IERT
60 GRAPHICS 0
70 SETCOLOR 2,12,8:REM GRUENER HINTERG
RUND
80 SETCOLOR 1,0,0:REM SCHWARZE ZEICHEN
90 FOR H=1 TO 5
100 ZSATZ(H)=(PEEK(742)-H*4)*256:REM A
NFANGSADRESSEN DER ZEICHENSAETZE
110 FOR I=ZSATZ(H) TO ZSATZ(H)+7
120 READ X
130 POKE I,X:REM SPEICHERN DER NEUEN Z
EICHEN
140 NEXT I
150 NEXT H
160 POSITION 0,0
170 LIST :REM BILDSCHIRM FUELLEN
180 FOR I=1 TO 5
190 POKE 756,INT(ZSATZ(I)/256)
200 IF I=2 THEN GOSUB 9000
210 FOR ZEIT=1 TO 15:NEXT ZEIT
220 NEXT I
230 GOTO 180
8999 REM SCHRITTGERAEUSCH
9000 FOR Q=0 TO 3
9010 SOUND Q,255,0,4
9020 NEXT Q
9030 FOR Q=0 TO 3
9040 SOUND Q,0,0,0
9050 NEXT Q
9060 RETURN
```

Die DATA-Befehle in Zeile 10 bis 50 enthalten die Information zur Darstellung der fünf neuen Zeichen.

Der Offset der im Zeichensatz gespeicherten Zeichen

Wenn Sie einen neuen Zeichensatz erstellen, müssen Sie beachten, daß nicht immer ein Zusammenhang zwischen dem ATASCII-Wert eines Zeichen und seiner Position im Zeichensatz besteht. Im obigen Programm wurde zum Beispiel das Zeichen zur Darstellung eines Zwischenraumes in fünf verschiedenen Zeichensätzen ersetzt. Überall dort, wo sich vorher ein Zwischenraum befand, wird also jetzt eines der neuen Zeichen erscheinen. Mit Hilfe eines einzigen POKE-Befehls schalten Sie zwischen den verschiedenen Zeichensätzen um und können Sie so blitzschnell gegeneinander austauschen.
Wenn Sie sich Ihren eigenen Zeichensatz erstellen wollen, ist es also nicht damit getan, die Bitmuster für die neuen Zeichen aufzustellen. In Tabelle 9-1 finden Sie den Offset aller ATASCII-Zeichen in Bezug auf das erste Zeichen des Zeichensatzes.

Tabelle 9-1: Offset der ATASCII-Zeichen in Bezug auf das erste Zeichen des Zeichensatzes*

ATASCII-Wert	Offset**
0-31	64-95
32-95	0-64
96-127	96-127
128-159	192-223
160-223	128-191
224-255	224-255

* Zur Berechnung der Anfangsadresse des gewünschten Zeichens (dezimal) multiplizieren Sie den Offset mit acht und zählen zum Ergebnis 57344 hinzu.

** Den Offset der Anfangsadresse des gewünschten Zeichens in Bezug auf die Anfangsadresse des Zeichensatzes erhalten Sie, indem Sie den angegebenen Offset mit acht multiplizieren.

Festlegung der Anfangsadresse des neuen Zeichensatzes

Zunächst müssen Sie einen freien Speicherbereich finden. Dann ist noch eine weitere Bedingung einzuhalten: die Anfangsadresse des neuen Zeichensatzes, muß, wenn Sie in der Grafikbetriebsart 0 arbeiten, ein Vielfaches von 1024 sein. Sollten Sie in der Grafikbetriebsart 1 oder 2 arbeiten, dann muß die Anfangsadresse durch 512 teilbar sein. In Speicherzelle 742 befindet sich die höchste dem Anwender zur Verfügung stehende Speicheradresse. Eine mögliche Anfangsadresse für Ihren eigenen Zeichensatz erhalten Sie, indem Sie von der in Speicherzelle 742 befindlichen Zahl 4 abziehen. Das gilt für die Grafikbetriebsart 0. Wenn Sie in der Grafikbetriebsart 2 arbeiten, müssen Sie stattdessen 2 abziehen. Das Ergebnis ist dann noch mit 256 zu multiplizieren. Genauso wurden auch die Anfangsadressen der fünf Zeichensätze im obigen Programm festgelegt.

Benutzung der Display-List

Die Verwaltung des Bildschirmes übernimmt im ATARI-Computer ein spezieller Mikroprozessor namens ANTIC. ANTIC besitzt, ähnlich wie der 6502-Mikroprozessor, einen eigenen Befehlssatz. Dieser Befehlssatz besteht hauptsächlich aus Anweisungen, mit deren Hilfe festgelegt wird, in welcher Form (welche Grafikbetriebsart) die Anzeige auf dem Bildschirm erfolgen soll. Sie können nun selbst Programme schreiben und Bildschirmformate

erstellen, die Sie vom BASIC aus nicht direkt aufrufen können. Ein solches Programm nennt man eine Display List.

Jedesmal, wenn Sie im BASIC einen GRAPHICS-Befehl geben, erstellt das Operating System eine Display List. ANTIC führt die in der Display List aufgeführten Anweisungen aus. Mit Hilfe dieser Anweisungen wird ANTIC mitgeteilt, ob es sich bei den im Bildschirmspeicher befindlichen Daten um Text oder Grafik handelt. ANTIC bearbeitet eine Anweisung nach der anderen und gibt das Ergebnis an einen anderen Prozessor (CTIA) weiter. ANTIC besitzt einen Programmzähler (das sogenannte Befehlsregister), ein Datenregister (den sogenannten Speicherzähler) und einige zusätzliche Register, die für interne Zwecke benötigt werden.
Mit jeder Anweisung der Display List kann die Grafikbetriebsart gewechselt werden. Es ist zum Beispiel möglich, auf einen Bildschirm zuerst fünf Textzeilen der Grafikbetriebsart 0, dann 60 Zeilen hochauflösende Grafik und schließlich noch einige Textzeilen der Grafikbetriebsart 1 oder 2 zu bringen. Sie können also den Bildschirm in mehrere horizontale Streifen unterteilen und für jeden Streifen eine andere Grafikbetriebsart wählen.

Die Arbeitsweise von ANTIC

Mit Hilfe der folgenden sieben Schritte wird sehr vereinfacht dargestellt, wie ANTIC bei der Abarbeitung der Display List vorgeht:

1. Holen einer Anweisung und Ablegen derselben im Befehlsregister.
2. Mit Hilfe der Anweisung wird die Grafikbetriebsart festgelegt. Dementsprechend betrachtet ANTIC den Inhalt des Bildschirmspeichers entweder als Text oder als Grafik.
3. Wenn Text dargestellt werden soll, liest ANTIC ein Zeichen aus dem Bildschirmspeicher, lokalisiert es im Zeichensatz und benutzt das dort gespeicherte Bitmuster, um das Zeichen auf dem Bildschirm darzustellen.
4. Der Inhalt des Bildschirmspeichers wird direkt auf den Bildschirm gebracht, wenn es sich bei den gespeicherten Daten um Grafik handelt.
5. Erhöhen des Zählers zur Adressierung der Display List. Damit kann die nächste Anweisung der Display List bearbeitet werden.
6. Erhöhen des Speicherzählers um die Anzahl von Bytes, die vorher aus dem Bildschirmspeicher gelesen wurden.
7. Sprung nach Schritt 1.

Darstellung der von ANTIC gelieferten Daten auf dem Bildschirm

ANTIC arbeitet die Display List kontinuierlich ab, das heißt, ANTIC liest die in der Display List befindlichen Anweisungen, verarbeitet den Inhalt des Bildschirmspeichers entsprechend, übergibt das Ergebnis an den CTIA (Baustein für die Erzeugung des Videosignals) und beginnt den ganzen Vorgang wieder von neuem. Der Elektronenstrahl des Fernsehgerätes bewegt sich unterdessen in waagrechten Linien von links nach rechts über den Bildschirm (Bild 9-2). Sobald das Ende des Bildschirmes erreicht ist, springt der Strahl wieder an den Anfang zurück.
Die waagrechten Linien nennt man die Bildschirmzeilen. Der ATARI-Computer erzeugt ein Videosignal mit 262 Bildschirmzeilen. Wenn das Ende einer Zeile erreicht ist, wird der Elektronenstrahl aus- und am Anfang der nächsten Zeile wieder eingeschaltet. Wenn das Ende der letzten Bildschirmzeile erreicht worden ist, wird der Elektronenstrahl für einen etwas längeren Zeitraum abgeschaltet. Er beginnt dann, sobald das nächste Bild dargestellt werden kann, wieder in der oberen linken Ecke des Bildschirms mit der ersten Bildschirmzeile. Diesen Vorgang nennt man die Vertikalaustastung.
Der ATARI-Computer bestimmt den Inhalt jeder einzelen Bildschirmzeile. Es sind jedoch

nicht alle 262 Zeilen, die vom Computer erzeugt werden, auf dem Bildschirm zu sehen. Aus technischen Gründen sind nur etwa 200 Bildschirmzeilen sichtbar. Damit es bei keinem Fernsehgerät Probleme gibt, benutzt der ATARI-Computer in allen Grafikbetriebsarten, die vom BASIC aus aufgerufen werden können, nur 192 Bildschirmzeilen zur Darstellung von Text oder Grafik. In jeder Grafikbetriebsart gibt es eine unterschiedliche Anzahl von Spalten und Zeilen. Je nach Wahl der Betriebsart werden zur Darstellung von einer dieser Zeilen 1 bis 16 Bildschirmzeilen benötigt.

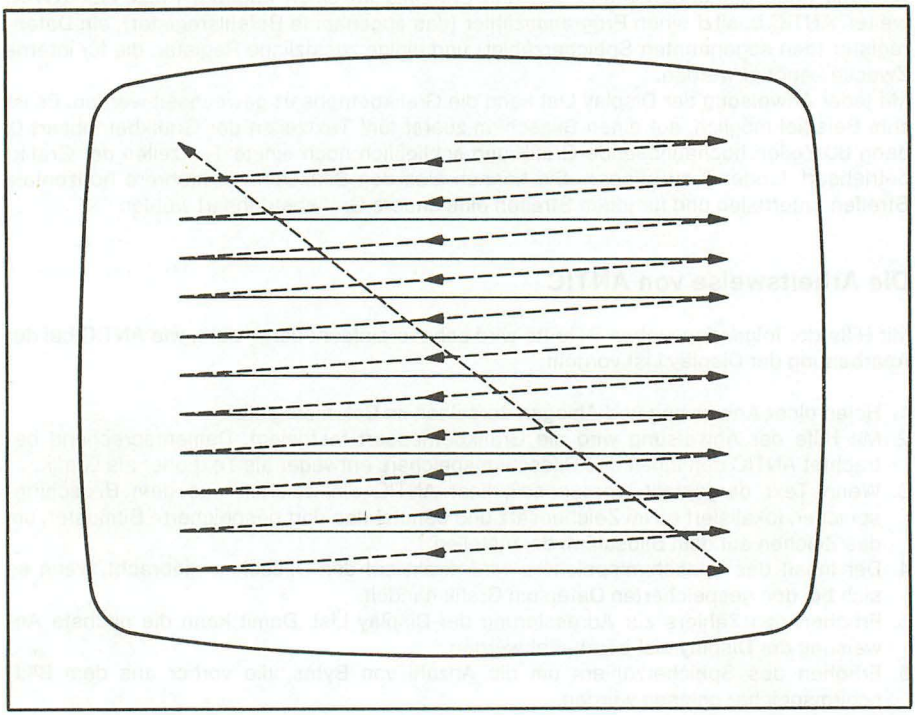


Bild 9-2: Die Bildschirmzeilen

Der Befehlssatz der Display-List

Bei der Display List gibt es Befehle zur

- Darstellung von Grafik
- Darstellung von Text
- Ausgabe von leeren Bildschirmzeilen
- Ausführung eines Sprunges

Außerdem gibt es noch die folgenden Zusatzfunktionen:

- Laden des Speicherzählers
- Verschieben des Bildschirminhaltes
- Erzeugen einer Unterbrechung (Interrupt)

Aufbau der Display List

Die Display List sollte auf eine ganz bestimmte Art aufgebaut sein. Um von den tatsächlich vorhandenen 262 Bildschirmzeilen auf 192 zu kommen, müssen zunächst einige leere Bildschirmzeilen ausgegeben werden. Es gibt mehrere Befehle, die speziell dafür vorgesehen sind. Dann muß der Speicherzähler mit der Anfangsadresse des Bildschirmspeichers, wo sich der Text oder die Grafik befindet, die auf dem Bildschirm dargestellt werden soll, geladen werden. Darauf folgen die Befehle zur Darstellung der gewünschten Grafikbetriebsart(en). Abschließend muß in der Display List ein Sprungbefehl stehen, bei dessen Auftreten ANTIC zurück an den Anfang der Display List springt. Einige andere Fälle, in denen ein Sprungbefehl gegeben werden muß, werden später besprochen.

Ausgabe von leeren Bildschirmzeilen

Obwohl es insgesamt acht Befehle zur Ausgabe von leeren Bildschirmzeilen gibt (siehe Tabelle 9-2), wird meistens nur der Befehl zur Darstellung von acht Leerzeilen verwendet (Kode 112 dezimal, 70 hexadezimal). Dieser Befehl steht gewöhnlich am Anfang der Display List.

Laden des Speicherzählers

Zum Laden des Speicherzählers gibt es keinen eigenen Befehl. Jeder Befehl kann jedoch so umfunktioniert werden, daß bei seiner Ausführung gleichzeitig der Speicherzähler geladen wird. Dazu müssen Sie zu einem Befehlscode 64 (40 hexadezimal) hinzuzählen. Bei der Ausführung dieses Befehls wird dann zunächst der Speicherzähler mit den zwei Bytes geladen, die direkt auf diese Anweisung folgen. Erst danach wird der umfunktionierte Befehl selbst ausgeführt. Jeder Befehl der Display List kann auf diese Weise modifiziert werden.

Sprungbefehle

ANTIC unterscheidet zwei verschiedene Arten von Sprüngen. Der erste ist ein unbedingter Sprung, bei dessen Ausführung der Zähler zur Adressierung der Display List mit einer neuen Adresse geladen wird und die Abarbeitung der Display List an diesem Punkt fortgesetzt wird. Der zweite Sprungbefehl sollte immer am Ende einer Display List stehen. Bei der Ausführung dieses Befehls wird auf die nächste Vertikalaustastung gewartet. Nachdem die letzte Bildschirmzeile auf den Bildschirm geschrieben worden ist, wird eine 1400 Mikrosekunden lange Pause eingelegt, während der der Elektronenstrahl an die linke obere Ecke des Bildschirms zurückkehren kann. Wenn stattdessen am Ende der Display List ein normaler Sprungbefehl verwendet würde, dann wären der Computer und das Fernsehgerät nicht mehr miteinander synchronisiert, was eine schlechte Bildqualität zur Folge hätte.

Gegenüberstellung der ANTIC-Befehle und der Grafikbetriebsarten des BASIC

Wie bereits erwähnt, können Sie mit Hilfe von ANTIC mehrere Grafikbetriebsarten gleichzeitig darstellen. Außerdem bietet Ihnen ANTIC einige zusätzliche Betriebsarten, die vom BASIC aus überhaupt nicht aufgerufen werden können. In den ersten drei Spalten von Tabelle 9-2 finden Sie neben den ANTIC-Befehlen die entsprechende Grafikbetriebsart des BASIC. Wie Sie sehen, gibt es für die ANTIC-Befehle 3, 4, 5, 12 und 14 im BASIC

nichts entsprechendes. Es ist also nicht möglich, diese Betriebsarten mit Hilfe des GRAPHICS-Befehls zu wählen. Zur Veranschaulichung der nun folgenden Erläuterungen dient das in Bild 9-3 aufgeführte Beispiel einer Display List zur Darstellung der Grafikbetriebsart 2 (ohne Textfenster).

Tabelle 9-2: Befehle zum Erstellen einer Display List

	Befehlskode		Grafikbetriebsart im BASIC	Horizontale Auflösung in Punkten	Bytes pro (Bildschirm)Zeile	Bildschirmzeilen pro Zeile	Bit pro Punkt	Auswahl der Farbe (D = Daten, CC = Farbinformation)
	Dezimal	Hex						
Ausgabe von leeren Bildschirmzeilen		00	–	–	–	1	–	
	16	10	–	–	–	2	–	
	32	20	–	–	–	3	–	
	48	30	–	–	–	4	–	
	64	40	–	–	–	5	–	
	80	50	–	–	–	6	–	
	96	60	–	–	–	7	–	
	112	70	–	–	–	8	–	
Textdarstellung	2	02	0	40	40	8	8	DDDDDDDD*
	3	03	–	40	40	10	8	
	4	04	–	40	40	8	8	CC CC CC CC†
	5	05	–	40	40	16	8	
	6	06	1	20	20	8	8	CC DDDDDD *†
	7	07	2	20	20	16	8	
Grafikdarstellung	8	08	3	40	10	8	2	CC†
	9	09	4	80	10	4	1	C C = 0 – Hintergrund C = 1 – Register 0
	10	0A	5	80	20	4	2	CC† Siehe Befehlskodes 6, 7, 8
	11	0B	6	160	20	2	1	Siehe Befehlskode 9
	12	0C	–	160	20	1	1	
	13	0D	7	160	40	1	2	CC† Siehe Befehlskodes 6, 7, 8
	14	0E	–	160	40	8	2	
	15	0F	8	320	40	16	1	D*

* D = 0 – Register 2
D = 1 – Register 1

† CC = 00 – Register 0 CC = 10 – Register 2
CC = 01 – Register 1 CC = 11 – Register 3

Mit Hilfe der ersten drei Bytes mit der Codezahl 112 werden, wie Sie der Tabelle 9-2 entnehmen können, jeweils 8, insgesamt also 24 Leerzeilen ausgegeben. Die Leerzeilen haben dieselbe Farbe wie der Hintergrund. Diese drei Befehl sollten am Anfang von jeder Display List stehen. Wenn Sie sie weglassen, wäre der Bildschirmanfang bei manchen Fernsehgeräten möglicherweise nicht mehr zu sehen. Der nächste Befehl hat eine Länge von drei Bytes. Es handelt sich hierbei um einen etwas modifizierten Befehl zur Darstellung

Byte	Befehlskode (dezimal)	
0	112	} Mit Hilfe dieser Befehle
1	112	} werden 24 Leerzeilen auf den
2	112	} Bildschirm gebracht
3	71	Befehl zum Laden des Speicherzählers und zur Darstellung der Grafikbetriebsart 2
4	59	(3B) Anfangsadresse des Bildschirm-
5	80	(50) speichers (503B hexadezimal)
6	7	
7	7	
8	7	
9	7	
10	7	
11	7	
12	7	
13	7	
14	7	
15	7	
16	7	
17	65	Sprung nach dem Auftreten einer Vertikalaustastung
18	0	(00) Anfangsadresse der
19	79	(4F) Display List

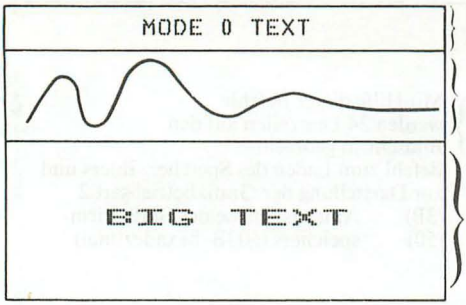
Anmerkung: Diese Display List entspricht der, die das Operating System bei Ausführung des Graphics 2-Befehls erstellt.

Bild 9-3: Beispiel einer Display List

von einer Zeile der ANTIC-Betriebsart 7 (im BASIC Grafikbetriebsart 2). Zu der Kodezahl dieses Befehls (7) wurde 64 (40 hexadezimal) addiert. Bei der Ausführung des Befehls wird dann zusätzlich der Speicherzähler mit den zwei Bytes der Display List geladen, die auf diese Kodezahl folgen. Jeder ANTIC-Befehl kann durch Addition von 64 auf diese Weise modifiziert beziehungsweise erweitert werden. In unserem Beispiel wird ANTIC mitgeteilt, daß die auf dem Bildschirm darzustellenden Daten von nun an ab der Adresse 20539 (503B hexadezimal) zu lesen sind. Bei der Angabe dieser Bildschirmspeicheradresse steht das niederwertige Byte wie üblich an erster Stelle.
Mit Hilfe der nächsten 11 Befehle werden 11 weitere Zeilen der Grafikbetriebsart 2 auf den Bildschirm gebracht. Am Schluß der Display List steht noch einmal ein 3-Byte-Befehl. Es handelt sich hierbei um einen Sprungbefehl, gefolgt von der Adresse, zu der gesprungen werden soll. An dieser Adresse wird dann die Ausführung der Display List fortgesetzt.
Die Display List ist im Grunde ziemlich einfach aufgebaut. Eine eigene Display List von Hand einzugeben ist allerdings trotzdem nicht gerade einfach, weil doch einigermaßen viele Regeln zu beachten sind.

Aufbau einer eigenen Display List

Nehmen Sie zum Beispiel einmal an, Sie bräuchten zu irgendeinem Zweck die folgende Bildschirmaufteilung:
An Stelle der Verschiebung kann auch der Zwischenraum entfernt und in diesem Satz die „die folgende Bildschirmaufteilung:" durch „die auf der nächsten Seite gezeigte Bildschirmaufteilung."
Zusätzlich zu den 24 Leerzeilen am Anfang der Display List sollten nach Möglichkeit 192



Normaler Text
(2 Zeilen x 8 Bildschirmzeilen = 16)

Grafikbetriebsart 8
(80 Zeilen x 1 Bildschirmzeile = 80)

Vergrößerter Text
(6 Zeilen x 16 Bildschirmzeilen = 96)

Summe 192

weitere Zeilen zur Darstellung von Text oder Grafik auf den Bildschirm gebracht werden. Zur Berechnung der Zeilenzahl sehen Sie in Tabelle 9-2 unter der Spalte mit der Bezeichnung "Bildschirmzeilen pro Zeile" nach. Im obigen Beispiel wurden exakt 192 Bildschirmzeilen ermittelt. Achten Sie darauf, daß Sie beim Aufbau der Display List keinen Fehler machen. ANTIC bringt soviele Bildschirmzeilen auf den Bildschirm, wie in der Display List angegeben sind. Wenn das zu viele sind, dann beginnt das Bild auf Ihrem Fernsehgerät zu laufen.

Ablegen der Display List im Speicher

Beim Ablegen der Display List im Speicher sind einige Dinge zu beachten. Die Display List selber darf eine 1K-Grenze im Speicher (eine durch 1024 teilbare Speicheradresse) nicht überschreiten, weil der Zähler zur Adressierung der Display List kein 16-Bit-Register ist. Eine solche Grenze kann jedoch, wie der folgende Ausschnitt einer Display List zeigt, übersprungen werden:

Inhalt der Display List (dezimal)			Adresse im RAM
14			1020
61		Sprung auf Adresse 1025	1021
1	(01)	(1024 überspringen)	1022
64	(40)		1023
[keine Anweisung]			1024
14			1025

Der Speicherzähler ist ebenfalls kein 16-Bit-Register. Zusätzlich zu dem am Anfang der Display List befindlichen Befehl zum Laden des Speicherzählers muß aus diesem Grund immer dann ein solcher Befehl auftauchen, wenn der Speicherzähler eine 4K-Grenze überschreiten soll. Das Ganze sieht also ähnlich aus wie bei dem Zähler zur Adressierung der Display List. Falls Ihre Display List sehr viel Bildschirmspeicher benötigt (zum Beispiel bei Darstellung der ANTIC-Betriebsart 15 = Grafikbetriebsart 8 im BASIC) und es somit nicht zu vermeiden ist, daß der Speicherzähler eine 4K-Grenze überschreitet, dann müssen Sie wie in dem auf der nächsten Seite gezeigten Beispiel vorgehen, um diese Grenze zu überspringen.

Als nächstes stellt sich nun die Frage, an welcher Stelle des Speichers die Display List abgelegt werden soll. Es ist natürlich möglich, die durch das Operating System erstellte Display List zu ersetzen. Die Display List kann jedoch auch in Page (= Seite) 6 gespeichert werden (Speicherbereich von Adresse 1536 = 6 x 256 bis 1791 = 7 x 256 – 1 beziehungsweise 600 bis 6FF hexadezimal). Dieser Bereich wird vom BASIC normalerweise nicht benutzt.

Wenn nicht gerade sehr wenig freier Speicherplatz vorhanden ist, sollten Sie die be-

stehende Display List jedoch nicht überschreiben. In Page 6 steht Ihnen genügend Platz zur Verfügung. Nachdem die neue Display List mit Hilfe des POKE-Befehls oder des Unterprogrammes zur Speicherung einer Display List (Bild 9-4) im Speicher abgelegt wurde, muß die Anfangsadresse dieser neuen Display List in den Speicherzellen 560 und 561 (230 und 231 hexadezimal) gespeichert werden. Wenn ANTIC während dieses Vorganges aktiv ist, kann es vorkommen, daß er eine falsche Anfangsadresse erkennt. Um das zu verhindern, schalten Sie ANTIC ab, bevor Sie die neue Anfangsadresse der Display List in den Speicherzellen 560 und 561 (230 und 231 hexadezimal) ablegen. Dazu brauchen Sie nur den Befehl POKE 559,0 zu geben. Dann speichern Sie die neue Anfangsadresse und schalten ANTIC mit POKE 559,34 wieder ein. Genauso wurde auch in dem in Bild 9-4 dargestellten Programm verfahren (Zeile 150 bis 190).

Unterprogramm zum Speichern einer Display List

Wenn Sie auf dem Bildschirm mehrere ANTIC-Betriebsarten gleichzeitig darstellen wollen, dann müssen Sie dafür ziemlich viele Berechnungen anstellen. Der ganze Vorgang wäre auf jeden Fall ziemlich zeitaufwendig. Das in Bild 9-4 aufgeführte Programm führt alle notwendigen Berechnungen durch und erstellt die gewünschte Display List. Sie brauchen nur noch die Anfangsadresse der Display List festzulegen und einige DATA-Befehle im Programm zu ändern. Mit dem GRAPHICS-Befehl in Zeile 20 müssen Sie die Grafikbetriebsart wählen, die von allen in Ihrer Display List vorkommenden Betriebsarten den meisten Speicherplatz belegt.
Mit Hilfe der DATA-Befehle bestimmen Sie, welche ANTIC-Betriebsarten (also nicht die Grafikbetriebsarten des BASIC) auf dem Bildschirm dargestellt werden sollen und wieviele Zeilen von jeder Betriebsart dargestellt werden sollen. Im obigen Beispiel sind das 3 Zeilen der BASIC-Grafikbetriebsart 0, jeweils eine Zeile der BASIC-Grafikbetriebsarten 1 und 2 und 144 Zeilen der BASIC-Grafikbetriebsart 8 (Zeile 40 bis 80). Das ergibt insgesamt die übliche Zahl von 192 Bildschirmzeilen. Die Adresse, ab der die Display List gespeichert werden soll, wird in Zeile 140 der Variablen LST zugewiesen. Im obigen Beispiel ist das die Adresse 1536 (600 hexadezimal). Das hoch- und das niederwertige Byte dieser Adresse muß später noch einmal in den Speicherzellen 561 und 560 (231 und 230 hexadezimal) abgelegt werden (Zeile 170 und 180).
In den Zeilen 200, 270, 310 und 350 wird der Variablen X die Nummer des Bildschirmabschnittes zugewiesen, in den geschrieben oder gezeichnet werden soll. Eine 0 steht dabei für den ersten Bildschirmabschnitt, der mit Hilfe der DATA-Befehle festgelegt wurde. Durch das Ablegen dieser Zahl in X und dem darauf folgenden GOSUB 430-Befehl wird vermieden, das bei der Adressierung des Bildschirmspeichers Fehler auftreten. Bei dieser Vorgehensweise wird dem Operating System simuliert, daß sich der angesprochene Bildschirmabschnitt am Anfang des Bildschirmes befindet. Die Zeilen 430 bis 450 und 1700 bis 2080 können Sie auch in eigenen Programmen als Unterprogramm verwenden.

```
1 REM BEISPIEL FUER EINE DISPLAY LIST
MIT EINEM UNTERPROGRAMM ZUM ERSTELLEN
EINER DISPLAY LIST
10 DIM ANFBILD(5):REM ANZAHL DER STREI
FEN, IN DIE DER BILDSCHIRM UNTERTEILT
WERDEN SOLL=5
20 GRAPHICS 8:REM SPEICHERPLATZ FUER D
EN BILDSCHIRMSPEICHER RESERVIEREN
30 SETCOLOR 2,0,0
40 DATA 2,3
50 DATA 6,1
60 DATA 7,1
70 DATA 15,144
80 DATA -1,0
87 REM MIT HILFE DER ERSTEN VIER DATA-
BEFEHLE WIRD DIE
88 REM AUFTEILUNG DES BILDSCHIRMES FES
TGELEGT. SIE SIEHT
89 REM IN DIESEM BEISPIEL FOLGENDERMAS
SEN AUS:
90 REM 3 ZEILEN DER GRAFIKBETRIEBSART
0 (BASIC),
100 REM 1 ZEILE DER GRAFIKBETRIEBSART
1 (BASIC),
110 REM 1 ZEILE DER GRAFIKBETRIEBSART
2 (BASIC)
120 REM UND 144 ZEILEN DER GRAFIKBETRI
EBSART 8 (BASIC)
130 REM DER LETZTE DATA-BEFEHL KENNZEI
CHNET DAS ENDE DER AUFSTELLUNG
140 LST=1536:REM FREIER RAM-BEREICH; G
UT GEEIGNET ZUR SPEICHERUNG DER DISPLA
Y LIST
150 GOSUB 1700:ERSTELLEN DER DISPLAY L
IST
160 POKE 559,0:REM ANTIC ABSCHALTEN
170 POKE 560,0:REM SPEICHERN DER NEUEN
 ANFANGSADRESSE DER DISPLAY LIST
180 POKE 561,6
190 POKE 559,34:ANTIC WIEDER EINSCHALT
EN
200 X=0:GOSUB 430:REM 1. BILDSCHIRMABS
CHNITT
210 POKE 87,0:REM GRAFIKBETRIEBSART 0
SIMULIEREN
220 POKE 752,1:CURSOR ABSCHALTEN
230 POSITION 5,0
240 PRINT #6;"WIE SIE SEHEN, KOENNEN S
IE MIT"
250 POSITION 2,1
260 PRINT #6;"HILFE DER DISPLAY LIST V
ERSCHIEDENE"
270 X=1:GOSUB 430:REM 2. BILDSCHIRMABS
CHNITT
```

Bild 9-4: Programm zum Erstellen einer Display List

```
280 POKE 87,1:REM GRAFIKBETRIEBSART 1
SIMULIEREN
290 POSITION 0,0
300 PRINT #6;" GRAFIKBETRIEBSARTEN"
310 X=2:GOSUB 430:REM 3. BILDSCHIRMABS
CHNITT
320 POKE 87,2:REM GRAFIKBETRIEBSART 2
SIMULIEREN
330 POSITION 5,0
340 PRINT #6;"VERMISCHEN"
350 X=3:GOSUB 430:REM 4. UND LETZTER B
ILDSCHIRMABSCHNITT
360 POKE 87,8:REM GRAFIKBETRIEBSART 8
SIMULIEREN
370 COLOR 1
380 PLOT 0,0
390 DRAWTO 319,143
400 PLOT 319,0
410 DRAWTO 0,143
420 STOP
430 POKE 88,ANFBILD(X)-(INT(ANFBILD(X)
/256)*256)
450 POKE 89,INT(ANFBILD(X)/256):RETURN
1690 REM ****************************
1691 REM *ERSTELLEN DER DISPLAY LIST*
1692 REM *==========================*
1693 REM *LST MUSS DIE ANFANGSADRES-*
1694 REM *SE DER NEUEN DISPLAY LIST *
1695 REM *  ENTHALTEN. BEI UEBER-  *
1696 REM * TEN EINER 1K-GRENZE WIRD *
1697 REM *AUTOMATISCH EIN JMP-BEFEHL*
1698 REM *     EINGEFUEGT.      *
1699 REM ****************************
1700 ABSCHNITT=1
1710 DISP=LST
1720 ANFBILD(0)=PEEK(88)+PEEK(89)*256:
REM ANFANGSADRESSE DES BILDSCHIRMSPEIC
HERS
1730 GRENZE=INT((LST/1024+1)*1024):REM
 NAECHSTE 1K-GRENZE
1740 GRENZE2=INT(ANFBILD(0)/4096+1)*40
96:REM NAECHSTE 4K-GRENZE
1750 FOR X=DISP TO DISP+2
1760 POKE X,112:REM DREI MAL 8 LEERE B
ILDSCHIRMZEILEN AM ANFANG DER DISPLAY
LIST
1770 DISP=DISP+1
1780 NEXT X
1790 ANFBILD(ABSCHNITT)=ANFBILD(ABSCHN
ITT-1):REM SETZEN DER ANFANGSADRESSE F
UER DIESEN ABSCHNITT
1800 READ BETRIEBSART,WDH
1810 IF BETRIEBSART<0 THEN KODE=65:ADD
R=LST:GOSUB 2030:RETURN
```

Bild 9-4: Programm zum Erstellen einer Display List (Fortsetzung)

```
1820 BYTE=40:ANZAHL DER BYTES ZUR DARS
TELLUNG VON EINER ZEILE
1830 IF BETRIEBSART>=6 AND BETRIEBSART
<=12 THEN BYTE=20
1840 IF BETRIEBSART=8 OR BETRIEBSART=9
 THEN BYTE=10
1850 FOR X=1 TO WDH
1860 IF DISP<>GRENZE-3 THEN 1900:REM U
EBERPRUEFEN, OB EINE 1K-GRENZE UEBERSC
HRITTEN WIRD
1870 KODE=1:ADDR=GRENZE:GOSUB 2030:REM
 EINFUEGEN EINES SPRUNGBEFEHLS
1880 DISP=DISP+1:GRENZE=GRENZE+1023
1890 GOTO 1900
1900 POKE DISP,BETRIEBSART:REM SPEICHE
RN DER BETRIEBSART
1910 REM MUSS DER SPEICHERZAEHLERS GEL
ADEN WERDEN?
1920 IF (ABSCHNITT<>1 OR X<>1) AND (GR
ENZE2-ANFBILD(ABSCHNITT)>=BYTE) THEN 1
980
1930 REM SPEICHERZAEHLER WIRD GELADEN!
1940 KODE=BETRIEBSART+64
1950 ADDR=ANFBILD(ABSCHNITT)
1960 IF GRENZE2-ANFBILD(ABSCHNITT)<BYT
E THEN GRENZE2=GRENZE2+4096:ADDR=ADDR+
BYTE
1970 GOSUB 2030
1980 DISP=DISP+1:REM ERHOEHEN DES ADRE
SSENZAEHLERS FUER DIE DISPLAY LIST
1990 ANFBILD(ABSCHNITT)=ANFBILD(ABSCHN
ITT)+BYTE
2000 NEXT X
2010 ABSCHNITT=ABSCHNITT+1
2020 GOTO 1790:REM LESEN DER INFORMATI
ON FUER DEN NAECHSTEN BILDSCHIRMABSCHN
ITT
2030 POKE DISP,KODE:REM EINFUEGEN EINE
S SPRUNGBEFEHLS
2040 DISP=DISP+1
2050 POKE DISP,ADDR-(INT(ADDR/256)*256
)
2060 DISP=DISP+1
2070 POKE DISP,INT(ADDR/256)
2080 RETURN
```

Bild 9-4: Programm zum Erstellen einer Display List (Fortsetzung)

Unterbrechen der DISPLAY LIST (INTERRUPT)

Die Tatsache, daß in eine Display List ein Interrupt eingefügt werden kann, mag zwar zunächst etwas außergewöhnlich erscheinen, aber dadurch ergeben sich völlig neue Möglichkeiten der Programmierung. Nachdem ein Befehl der Display List abgearbeitet wurde, holt ANTIC den nächsten Befehl. Wenn bei diesem Befehl bestimmte Bits gesetzt sind, er-

folgt ein Interrupt, das heißt, ANTIC übergibt die Kontrolle an ein spezielles Unterprogramm, dessen Ausführung nicht mehr als 18 Maschinenzyklen des 6502-Mikroprozessors dauern darf. Mit 18 Maschinenzyklen können Sie zwar keine großen Programme schreiben, aber diese Zeitspanne reicht auf jeden Fall aus, um den Inhalt eines Farb- oder eines anderen Registers zu ändern. Die nun folgenden Aufstellungen zeigen Ihnen, was Sie bei der Einfügung eines Interrupts beachten müssen.
Um einen Interrupt in eine Display List einzubauen, gehen Sie folgendermaßen vor:

1. Speichern Sie das Unterprogramm, das bei Auftreten eines Interrupts abgearbeitet werden soll, in einem Speicherbereich, auf den sonst nicht zugegriffen wird, zum Beispiel in Page 6 (Speicheradresse 1536 bis 1791 beziehungsweise 600 bis 6FF hexadezimal).
2. Modifizieren Sie die Bytes der Display List, vor deren Ausführung ein Interrupt erfolgen soll.
3. Speichern Sie mit Hilfe des POKE-Befehls in den Speicherzellen 512 und 513 (200 und 201 hexadezimal) die Anfangsadresse des Unterprogrammes, das bei Auftreten eines Interrupts abgearbeitet werden soll. (Bedenken Sie, daß nicht ANTIC, sondern der 6502-Mikroprozessor dieses Unterprogramm abarbeitet.)
4. Schalten Sie den Interrupt ein, indem Sie den folgenden Befehl geben:
   POKE 54286,192

Bei der Erstellung des Unterprogrammes selbst sollten die folgenden Punkte beachtet werden:

1. Speichern Sie den Inhalt aller Register, die in diesem Unterprogramm verwendet werden, im Stack des 6502.
2. Darauf folgt nun das Unterprogramm selbst. Es sollte so kurz wie möglich gehalten werden. Auf keinen Fall darf die Ausführung des ganzen Programmes mehr als 18 Maschinenzyklen in Anspruch nehmen. Anderenfalls wird ANTIC kein einwandfreies Videosignal mehr liefern.
3. Laden Sie nun wieder die Daten, die Sie vorher im Stack abgelegt hatten und weisen Sie sie den entsprechenden Registern zu.
4. An das Ende des Unterprogrammes stellen Sie einen RTI-Befehl (Return from Interrupt-=Rückkehr vom Unterprogramm in das Hauptprogramm). Die Abarbeitung der Display List wird dann dort fortgesetzt, wo sie vorher unterbrochen worden war.

Diese Schritte müssen in jedem Unterprogramm vorhanden sein, das beim Auftreten eines Interrupts abgearbeitet werden soll. Anderenfalls könnte der Inhalt von einigen Registern des 6502, die noch benötigt werden, zerstört werden (und damit eventuell auch Ihr Programm).

Beispiel eines Display List Interrupts

Das folgende Programm wird sicher sehr hilfreich sein, wenn Sie bisher noch keine Erfahrung bei der Programmierung in Assemblersprache hatten. Nehmen wir einmal an, in der Grafikbetriebsart 0 sollen in der oberen Hälfte des Bildschirms alle Zeichen normal und in der unteren Hälfte alle Zeichen auf dem Kopf stehend dargestellt werden.
Wenn alle Zeichen des Bildschirmes auf dem Kopf stehen sollten, bräuchten Sie nur den Befehl POKE 755,6 geben. Aber so ist die Sache nicht ganz so einfach. Sie erzeugen in der Display List einen Interrupt, nachdem der halbe Bildschirm dargestellt wurde. In dem entsprechenden Unterprogramm legen Sie in der Speicherzelle 54273 (D401 hexadezimal) eine 4 ab. Dafür werden auf keinen Fall mehr als 18 Maschinenzyklen benötigt. Das durch das Speichern einer 4 in der Speicherzelle 54273 gesetzte Bit bewirkt, daß von nun an bis zum Auftreten der nächsten Vertikalaustastung alle Zeichen auf dem Kopf stehend dargestellt werden. Wenn ANTIC wieder an den Anfang der Display List zurückspringt, wird

dieses Bit automatisch zurückgesetzt.
Im folgenden finden Sie das Listing des Assemblerprogrammes, das in Page 6 gespeichert wird:

```
0000          10  *=$600
0600 48       20  PHA         ;SPEICHERN DES AKKUMULATORINHALTS IM STACK
0601 A904     30  LDA #4
0603 8D01D4   40  STA $D401   ;DARSTELLUNG ALLER ZEICHEN AUF DEM KOPF STEHEND
0606 68       50  PLA         ;LADEN DES VORHER GESPEICHERTEN AKKUMULATORINHALTS
0607 40       60  RTI         ;ABARBEITUNG DER DISPLAY LIST FORTSETZEN
```

Mit Hilfe des folgenden BASIC-Programms wird die Display List geändert. Modifiziert wird der zur Darstellung der 12. Bildschirmzeile zuständige Befehl der Display List. Dadurch wird der gewünschte Effekt erzielt: alle Zeichen der unteren Bildschirmhälfte werden auf dem Kopf stehend dargestellt. Versuchen sie es selbst und geben Sie das Programm einmal ein. Der DATA-Befehl in Zeile 35 enthält das obige Unterprogramm:

```
5 GRAPHICS 0
10 DLST=PEEK(560)+PEEK(561)*256:REM AN
FANGSADRESSE DER DISPLAY LIST
20 POKE DLST+16,130:REM MODIFIKATION Z
UR EINFUEGUNG DES INTERRUPTS
30 FOR I=1536 TO 1543:REM ABLEGEN DES
UNTERPROGRAMMES AB ADRESSE $600
35 DATA 72,169,4,141,1,212,104,64
40 READ X
50 POKE I,X
60 NEXT I
65 POKE 512,0
66 POKE 513,6:REM ANFANGSADRESSE DES U
NTERPROGRAMMES
70 POKE 54286,192:REM INTERRUPT EINSCH
ALTEN
80 FOR I=0 TO 23
90 POSITION 0,I:? "RICHTIGHERUM";
100 NEXT I
```

Andere Adressen, die Sie bei einem Interrupt ändern können, finden Sie im ATARI Personal Computer System Hardware Manual. Dieses Manual ist über ATARI zu erhalten. Dort finden Sie eine sehr viel weitergehende Beschreibung des Display List Interrupts. Sie werden damit in der Lage sein, alle Möglichkeiten, die Ihnen dieser Interrupt bietet, voll auszuschöpfen.

Player-Missile Grafiken

Den meisten Leuten fällt es einigermaßen schwer, den Begriff "Player-Missile Grafik" vorurteilsfrei zu betrachten. Sie denken dann meist an Spiele oder ähnliche Dinge, die bei der normalen Programmierung keine große Bedeutung haben. Dabei handelt es sich bei den Playern (= Spielern) und den Missiles (= Ziele) eigentlich nur um Objekte irgendeiner Art, die sehr schnell über den Bildschirm bewegt werden können. Maximal vier Player können gleichzeitig dargestellt werden. Zu jedem Player gibt es ein eigenes Missile. In Bild 9-5 finden Sie einige einfache Beispiele , wie ein Player aussehen könnte.
Die Player sind der Höhe (vertikal) und in der Breite (horizontal) begrenzt. Die maximale

Höhe beträgt 256 Zeilen, die maximale Breite acht Bit. Ein Player kann sich über die volle Höhe des Bildschirmes erstrecken. Die Missiles entsprechen im wesentlichen den Playern, allerdings beträgt ihre maximale Breite lediglich 2 Bit.
Natürlich können die Player für Spiele verwendet werden, aber genausogut können sie auch feststehende Objekte darstellen. Bei der Dateneingabe könnte ein Player zum Beispiel als Cursor verwendet werden oder alle vier Player könnten als Bildschirmbegrenzung dienen. Es gibt drei Gründe, die für den Einsatz der Player-Missile-Grafik sprechen: die Unabhängigkeit von anderen Grafikfunktionen des Computers und die Möglichkeit, eine schnelle Bewegung und mehr Farben darzustellen.
Player-Missile-Grafiken sind vollkommen unabhängig von den anderen Grafikfunktionen des ATARI-Computers. Die Grafikbetriebsarten, die Sie normalerweise wählen können, nennt man Playfield-Grafiken. PRINT, PLOT, DRAWTO und alle anderen BASIC-Befehle beziehen sich auf diese Playfield-Grafiken. Bei den Player-Missile-Grafiken handelt es sich um fest vorgegebene Objekte. Einige Beispiele dafür finden Sie in Bild 9-5. Diese Objekte werden dem normalen Bildschirminhalt überlagert, unabhängig davon, welche Grafikbetriebsart gerade verwendet wird. Außerdem können sie die Grenzen der Playfield-Grafiken überschreiten und sowohl vor als auch hinter einer Playfield-Grafik dargestellt werden. Damit besteht die Möglichkeit, einen dreidimensionalen Eindruck zu erzeugen.
Player können auf dem Bildschirm sehr schnell bewegt werden, ohne daß dabei die Rechengeschwindigkeit nachteilig beeinflußt wird. Sehen Sie sich noch einmal die in Bild 9-5 dargestellten Objekte an und überlegen Sie sich, wie Sie diese Objekte unter Verwendung des PLOT- und des DRAWTO-Befehls zeichnen würden. Wenn Sie es an eine andere Stelle des Bildschirms bewegen wollten, müßten Sie es zunächst löschen, dann die Koordinaten der neuen Position berechnen und das Objekt an dieser Stelle neu zeichnen. Dafür wird enorm viel Rechenzeit benötigt, weil der 6502-Mikroprozessor all diese Aufgaben selbst übernehmen muß. Bei Verwendung der Player-Missile-Grafiken wird stattdessen mit Hilfe von ANTIC die Technik des Direct Memory Access (= direkter Zugriff auf den Speicher ohne Einschaltung des Mikroprozessors) oder kurz DMA angewendet. Es wird sehr viel weniger Rechenzeit zur Bewegung des Objekts benötigt, wodurch der 6502-Mikroprozessor für andere Aufgaben zur Verfügung steht. Zusammenfassend kann man sagen, daß bei Verwendung von Player-Missile-Grafiken der 6502-Mikroprozessor umgangen wird, während bei der Benutzung von Playfield-Grafiken das Operating System und damit auch der 6502-Mikroprozessor benötigt wird.

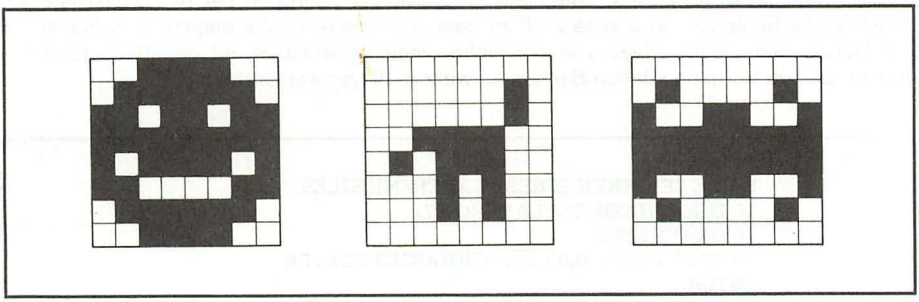


Bild 9-5: So könnte ein Player aussehen

Mit Hilfe der Player-Missile-Grafiken kann mehr Farbe auf den Bildschirm gebracht werden. Für jeden Player gibt es ein eigenes Farbregister, das von den für die Playfield-Grafik verwendeten Farbregistern völlig unabhängig ist. Ganz gleich, welche Grafikbetriebsart Sie gewählt haben, für jeden Player gibt es ein zusätzliches Farbregister. So ist es zum Beispiel möglich, daß Sie bei der Verwendung von Player-Missile-Grafiken in der Grafikbetriebsart 0 fünf Farben auf einmal auf den Bildschirm bringen können. In anderen Gra-

fikbetriebsarten können Sie so sogar neun Farben auf einmal darstellen.
Am Anfang mag es vielleicht etwas schwierig sein, die Funktionsweise der Player-Missile-Grafiken zu verstehen, weil es keine BASIC-Befehle gibt, mit denen sie direkt angesprochen werden können. Sie sind anders aufgebaut und werden anders benutzt (meistens mit Hilfe von Programmen, die in Maschinensprache geschrieben wurden) als Playfield-Grafiken. Die Programmerstellung wird dadurch natürlich nicht gerade erleichtert. Trotzdem sind fast alle der in diesem Abschnitt vorhandenen Programme zur Benutzung der Player-Missile-Grafiken zum größten Teil in BASIC geschrieben.

Organisation des Speichers bei Verwendung von Player-Missile-Grafiken

Wenn Sie Player-Missile-Grafiken in einem Programm verwenden wollen, dann müssen Sie eine Tabelle erstellen, in der die Form der Player festgehalten wird. Diese Tabelle legen Sie am besten an das obere Ende des RAM-Speichers. Es ist sehr unwahrscheinlich, daß dieser Speicherbereich anderweitig benutzt wird. Eine Einschränkung müssen Sie allerdings beachten, wenn Sie diese Tabelle anlegen: sie muß an einer 1024- oder 2048-Byte-Grenze beginnen, je nachdem wie groß die vertikale Auflösung der Player ist.

Festlegen der Form der Player

Jeder Player hat eine Breite von 8 Bit. Das gezeichnete Objekt (wie in Bild 9-5) muß in eine binäre Information umgeformt werden. Jeder ausgefüllte Teil des Gitters entspricht einer binären 1 und jeder nicht ausgefüllte Teil einer binären 0. In der Tabelle für die Player-Missile-Grafiken wird die Form von jedem Player durch eine Folge von Bytes festgelegt. Das in Bild 9-6 aufgeführte Programm wird sich beim Festlegen der Form der Player als sehr nützlich erweisen. Schließen Sie einen Joystick an Port 1 an und starten Sie das Programm. Der Bildschirm wird durch beidseitige Begrenzungen auf eine maximale Breite von acht Punkten beschränkt. Nachdem Sie einen Player gezeichnet haben, drükken Sie die RETURN-Taste. Daraufhin wird der Player in normaler Größe angezeigt. Falls Sie noch weitere Veränderungen vornehmen wollen, drücken Sie die Leertaste. Wenn der Cursor wieder auf dem Bildschirm erschienen ist, können Sie, wieder mit Hilfe des Joysticks, den Player verändern. Wenn Sie an Stelle der Leertaste die RETURN-Taste ein zweites Mal betätigen, wird zusätzlich zu dem in normaler Größe angezeigten Player die zur Darstellung dieses Players erforderliche binäre Information auf den Bildschirm gebracht. Diesen Vorgang können Sie nun für weitere Player wiederholen.

```
5 REM ZEICHNEN EINES PLAYER/MISSILES
10 DIM CURSOR(2),PLAYER(23,7)
20 GRAPHICS 3
30 SETCOLOR 2,0,0:REM SCHWARZES TEXTFE
NSTER
40 PRINT "STECKEN SIE DEN JOYSTICK IN
PORT 1"
45 PRINT "UND DRUECKEN SIE DEN AUSLOES
EKNOPF,"
50 GOSUB 610
```

Bild 9-6: Programm zum Zeichnen eines Player oder eines Missiles

```
60 PRINT "ZUM ZEICHNEN ODER LOESCHEN E
INES"
65 PRINT "PUNKTES DRUECKEN SIE DEN AUS
LOESE-"
70 PRINT "KNOPF. WENN DIE ZEICHNUNG FE
RTIG"
75 PRINT "IST, DRUECKEN SIE DIE RETURN
-TASTE.";
80 GOSUB 670:REM VORBEREITUNG ZUM ZEIC
HNEN
90 IF PEEK(764)=12 THEN GOTO 290:REM B
EENDEN, FALLS RETURN GEDRUECKT WURDE
100 GOSUB 770:REM STELLUNG DES JOYSTIC
KS BESTIMMEN
110 IF (OBEN AND (CURSOR(2)<=0)) OR (U
NTEN AND (CURSOR(2)>=23)) THEN GOTO 90
120 IF (LINKS AND (CURSOR(1)<=0)) OR (
RECHTS AND (CURSOR(1)>=7)) THEN GOTO 9
0
130 COLOR 1
140 IF PLAYER(CURSOR(2),CURSOR(1))=0 T
HEN COLOR 0
150 PLOT CURSOR(1)+16,CURSOR(2):REM CU
RSOR NEU POSITIONIEREN
160 CURSOR(2)=CURSOR(2)-OBEN
170 CURSOR(2)=CURSOR(2)+UNTEN
180 CURSOR(1)=CURSOR(1)-LINKS
190 CURSOR(1)=CURSOR(1)+RECHTS
200 COLOR 2
210 PLOT CURSOR(1)+16,CURSOR(2)
220 X1=STRIG(0):REM FALLS AUSLOESEKNOP
F GEDRUECKT WURDE; PUNKT ZEICHNEN ODER
 LOESCHEN
230 IF X1=1 THEN 90
240 PLAYER(CURSOR(2),CURSOR(1))=1-SGN(
PLAYER(CURSOR(2),CURSOR(1)))
250 COLOR 3
260 IF PLAYER(CURSOR(2),CURSOR(1))=0 T
HEN COLOR 0
270 PLOT CURSOR(1)+16,CURSOR(2)
280 GOTO 90
290 GRAPHICS 7:REM PLAYER NOCH EINMAL
IN HOEHERER AUFLOESUNG ZEICHNEN
300 POKE 764,0:REM LOESCHEN DES SPEICH
ERS FUER DAS LETZTE VON DER TASTATUR E
INGEGEBENE ZEICHEN
310 X2=2:X3=0
320 X1=76:Y1=20
330 GOSUB 910:REM PLAYER ZEICHNEN
350 PRINT:PRINT "DRUECKEN SIE DIE RETU
RN-TASTE, WENN"
352 PRINT "SIE DAS PROGRAMM BEENDEN UN
D DIE"
```

Bild 9-6: Programm zum Zeichnen eines Player oder eines Missiles (Fortsetzung)

```
354 PRINT "LEERTASTE, WENN SIE NOCH VE
RAEN-"
356 PRINT "DERUNGEN VORNEHMEN WOLLEN,"
;
360 IF PEEK(764)=12 THEN GOTO 440:REM
WENN RETURN GEDRUECKT WURDE; BINAERDAR
STELLUNG DES PLAYERS ANZEIGEN
370 IF PEEK(764)<>33 THEN 360:REM FALL
S DIE LEERTASTE NICHT GEDRUECKT WURDE,
 AUF TASTATURBETAETIGUNG WARTEN
380 GRAPHICS 3+16
390 GOSUB 740:REM VORBEREITEN ZUM VERA
ENDERN DES PLAYERS
400 X1=16:Y1=0
410 X2=1:X3=0
420 GOSUB 910
430 PRINT:GOTO 60
440 GRAPHICS 0:REM PLAYER UND BINARE D
ARSTELLUNG DES PLAYERS ANZEIGEN
450 FOR Y=0 TO 23
460 POSITION 0,Y
470 PRINT "BYTE ";Y;
480 X1=0
490 FOR X=0 TO 7
500 X2=PLAYER(Y,7-X)
510 IF X2=0 THEN 530
520 X1=X1+INT((X2*2)^X+0.01):REM JEDES
 BIT ENTSPRECHEND SEINEM STELLENWERT A
DDIEREN
530 NEXT X
540 POSITION 12,Y:PRINT X1;
550 NEXT Y
560 X1=24:Y1=0
570 X2=ASC(" "):X3=ASC(" ")
580 POKE 752,1
590 GOSUB 910
600 GOTO 590
610 REM DRUCK DES AUSLOESEKNOPFES ABWA
RTEN/KONTROLLTON ERZEUGEN
620 IF STRIG(0)=1 THEN 620
630 SOUND 0,50,10,4
640 FOR ZEIT=1 TO 10:NEXT ZEIT
650 SOUND 0,0,0,0
660 RETURN
665 REM VARIABLEN INITIALISIEREN UND B
ILDSCHIRM VORBEREITEN ZUM ZEICHNEN
670 CURSOR(1)=0
680 CURSOR(2)=0
690 FOR R=0 TO 23
700 FOR C=0 TO 7
710 PLAYER(R,C)=0
720 NEXT C
730 NEXT R
```

Bild 9-6: Programm zum Zeichnen eines Player oder eines Missiles (Fortsetzung)

```
740 COLOR 3
750 PLOT 15,0:DRAWTO 15,23:PLOT 24,0:D
RAWTO 24,23
760 GRAPHICS 3+48:RETURN
770 REM STELLUNG DES JOYSTICKS BESTIMM
EN
780 STELLUNG=STICK(0)
790 UNTEN=0
800 OBEN=0
810 LINKS=0
820 RECHTS=0
830 IF STELLUNG=15 THEN RETURN
840 IF STELLUNG=14 THEN OBEN=1
850 IF STELLUNG=7 THEN RECHTS=1
860 IF STELLUNG=13 THEN UNTEN=1
870 IF STELLUNG=11 THEN LINKS=1
880 GOSUB 630:REM KONTROLLTON ERZEUGEN
890 RETURN
900 REM PLAYER ZEICHNEN
910 FOR Y=0 TO 23
920 FOR X=0 TO 7
930 COLOR X2
940 IF PLAYER(Y,X)=0 THEN COLOR X3
950 PLOT X+X1,Y+Y1
960 NEXT X
970 NEXT Y
980 RETURN
```

Bild 9-6: Programm zum Zeichnen eines Player oder eines Missiles (Fortsetzung)

Darstellung der Player

Zur Festlegung der Form der Player benötigen Sie 128 oder 256 Bytes. Wenn nur 128 Bytes benutzt werden, erfolgt die Anzeige des Players auf dem Bildschirm wie in Bild 9-7 gezeigt. Jedes Byte belegt in vertikaler Richtung zwei Bildschirmzeilen. Bei der Verwendung von 256 Bytes belegt, wie Sie Bild 9-8 entnehmen können, jedes Byte des Players nur eine Bildschirmzeile.
Die vertikale Auflösung eines 256-Byte-Players ist doppelt so groß wie die eines 128-Byte-Players. Damit können also feinere Strukturen dargestellt werden. Sie sollten sich allerdings vorher überlegen, ob Sie die höhere Auflösung auch wirklich benötigen. Da alle Player die gleiche Auflösung haben müssen, können Sie durch diese Entscheidung ziemlich viel Speicherplatz einsparen. Die 128-Byte-Player nennt man auch Player mit einfacher Auflösung und die 256-Byte-Player Player mit doppelter Auflösung.

Die Tabelle zur Darstellung von Player-Missile-Grafiken

Die Anfangsadresse der Tabelle zur Darstellung von Player-Missile-Grafiken muß ohne Rest durch 1024 (für Player mit einfacher Auflösung) beziehungsweise durch 2048 (für Player mit doppelter Auflösung) teilbar sein. Sie können diese Tabelle an das Ende des Speichers legen. Dazu berechnen Sie zunächst mit Hilfe des BASIC-Befehls ?PEEK(106) x 256 die höchste Speicheradresse und ermitteln daraus die nächste 1024- beziehungsweise 2048-Byte-Grenze.

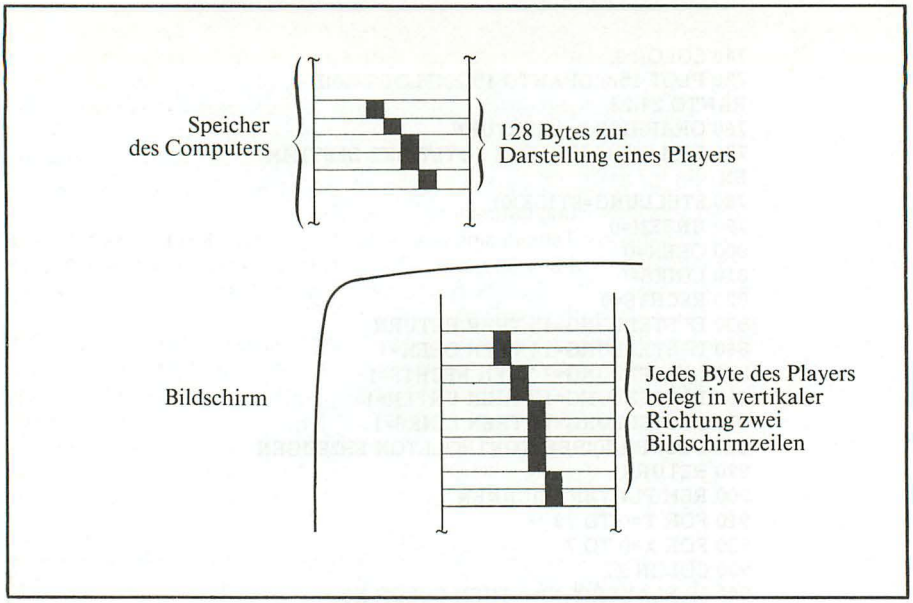


Bild 9-7: Player mit einfacher Auflösung

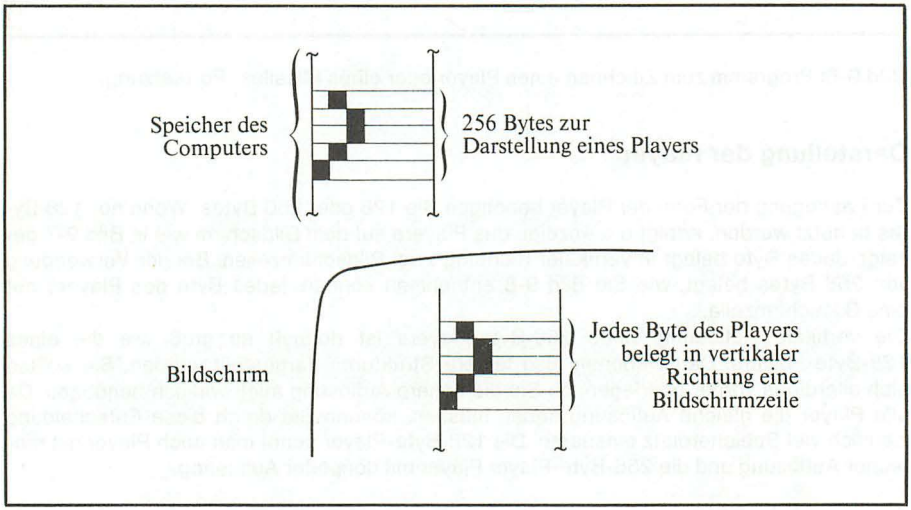
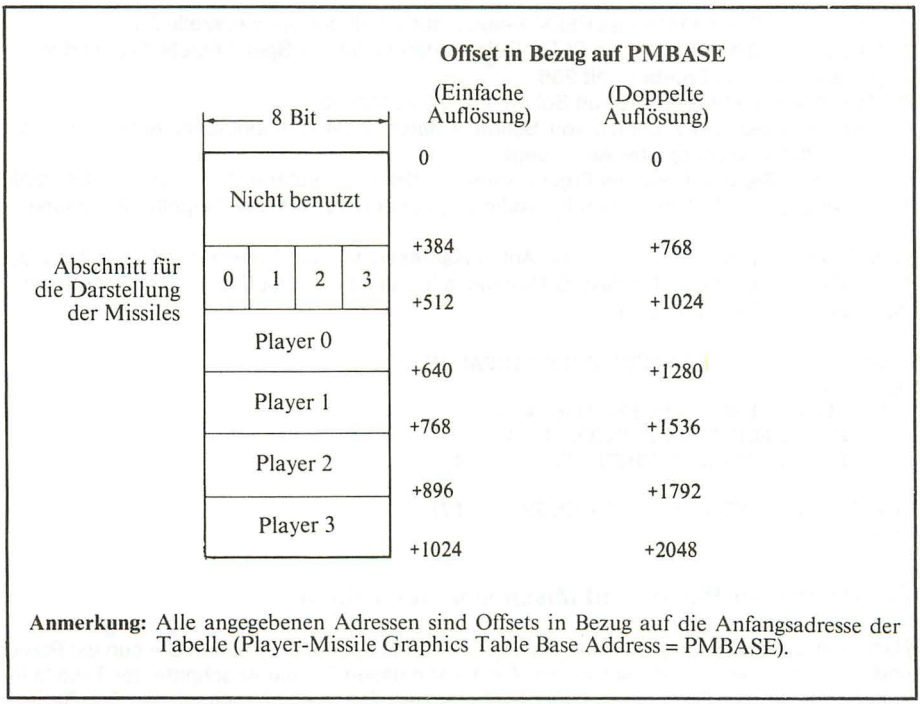


Bild 9-8: Player mit doppelter Auflösung

Organisation der Tabelle zur Darstellung der Player-Missile-Grafiken

Die Organisation der Tabelle zur Darstellung der Player-Missile-Grafiken finden Sie in Bild 9-9. Die Länge dieser Tabelle ist nicht variabel. Ganz gleich, wieviele Player Sie tatsächlich benötigen, die Tabelle ist immer 1024 beziehungsweise 2048 Bytes lang, damit genug

Platz für alle Player und Missiles vorhanden ist. Der erste Abschnitt der Tabelle wird nicht benutzt und steht für andere Zwecke zur Verfügung. Hier können Sie zum Beispiel die Information zur Darstellung eines fünften Players, gegen den Sie eventuell einen der anderen vier austauschen oder eine Display List speichern. Auf diesen ersten Abschnitt folgen fünf weitere, in denen die Form der Player und der Missiles festgelegt wird.
Im Abschnitt zur Speicherung der Missiles ist Platz für vier Missiles vorhanden, von denen jedes eine Breite von zwei Bit hat. Genauso wie die Player können auch die Missiles in einfacher oder in doppelter Auflösung dargestellt werden.
Die nächsten vier Abschnitte der Tabelle sind alle gleich lang. Jeder dieser Abschnitte ist für einen Player zuständig. In Bild 9-9 finden Sie die Offsets der einzelnen Abschnitte in Bezug auf die Anfangsadresse der Tabelle. Sie brauchen diese Offsets, wenn Sie einen Player oder ein Missile über den Bildschirm bewegen wollen.
Die Anfangsadresse der Tabelle zur Darstellung der Player-Missile-Grafiken müssen Sie in der Speicherzelle 106 (6A hexadezimal) speichern. Der Inhalt dieser Speicherzelle wird zur Berechnung der Anfangsadresse des Bildschirmspeichers für die Playfield-Grafiken benutzt. Dieser Bildschirmspeicher und die Tabelle zur Darstellung der Player-Missile-Grafiken würden sich überlappen, wenn Sie den Inhalt der Speicherzelle 106 nicht ändern würden, was zu Fehlfunktionen bei den Playfield-Grafiken oder bei den Player-Missile-Grafiken oder sogar bei beiden führen würde. Schlimmstenfalls könnte der Computer sogar in eine Endlosschleife geraten.



Anmerkung: Alle angegebenen Adressen sind Offsets in Bezug auf die Anfangsadresse der Tabelle (Player-Missile Graphics Table Base Address = PMBASE).

Bild 9-9: Organisation der Tabelle zur Darstellung der Player-Missile-Grafiken

Berechnen der Anfangsadresse der Tabelle zur Darstellung der Player-Missile-Grafiken

Wenn sich verschiedene Speicherbereiche überlappen, dann kann der ATARI-Computer

diesen Fehler nicht beheben. Stattdessen müssen Sie dafür sorgen, daß es keine Überlappungen gibt. Sie wissen bereits, daß die Tabelle zur Darstellung der Player-Missile-Grafiken an einer 1K- oder 2K-Grenze beginnen muß. Aber auch der Bildschirmspeicher für die Playfield-Grafiken unterliegt gewissen Einschränkungen. Wenn Sie die Tabelle für die Player-Missile-Grafiken an das Ende des Speichers legen, werden aus diesem Grund bei den Playfield-Grafiken Probleme auftreten. Verschiedene Bereiche des Bildschirmes können zum Beispiel nicht mehr beschrieben werden oder bei Verwendung eines PLOT-Befehls können bestimmte Koordinaten nicht angesprochen werden.
Diese Probleme treten nicht auf, wenn Sie die Lage des Bildschirmspeichers für die Playfield-Grafiken nicht verändern und die Tabelle für die Player-Missile-Grafiken vor diesen Bildschirmspeicher legen. Dabei können jedoch andere Probleme auftreten, wenn Sie im Verlauf eines Programmes die Grafikbetriebsart wechseln. Es kann vorkommen, daß dabei die gesamte Tabelle gelöscht wird. Das wäre zum Beispiel dann der Fall, wenn Sie von der Grafikbetriebsart 0 umschalten auf die Grafikbetriebsart 8. Dann würde tatsächlich die ganze Tabelle vollständig gelöscht werden. Sie können das jedoch verhindern, indem Sie in Ihrem Programm den GRAPHICS-Befehl ausfindig machen, für den der meiste Speicherplatz benötigt wird und dementsprechend die Anfangsadresse der Tabelle festlegen. Um die Anfangsadresse zu berechnen, wählen Sie am besten die in Ihrem Programm vorkommende Grafikbetriebsart, für die der meiste Speicherplatz benötigt wird und gehen dann folgendermaßen vor:

1. Bestimmen Sie mit Hilfe des PEEK-Befehls den Inhalt der Speicherzelle 560.
2. Bestimmen Sie mit Hilfe des PEEK-Befehls den Inhalt der Speicherzelle 561 und multiplizieren Sie das Ergebnis mit 256.
3. Zählen Sie die Ergebnisse von Schritt 1 und 2 zusammen.
4. Dividieren Sie das Ergebnis von Schritt 3 durch 1024 (bei einfacher Auflösung) oder durch 2048 (bei doppelter Auflösung).
5. Streichen Sie den Rest des Ergebnisses von Schritt 4, subtrahieren Sie 1 und multiplizieren Sie mit 1024 (bei einfacher Auflösung) oder mit 2048 (bei doppelter Auflösung).

Das Ergebnis von Schritt 5 ist die Anfangsadresse für die Tabelle zur Darstellung der Player-Missile-Grafiken. Ein BASIC-Programm für die oben aufgeführten fünf Schritte würde folgendermaßen aussehen:

```
1000 REM EINFACH=1 STEHT FUER EINFACHE
 AUFLOESUNG
1010 PMBASE=PEEK(560)+PEEK(561)*256
1020 IF EINFACH THEN DIVISOR=1024
1030 IF (NOT EINFACH) THEN DIVISOR=204
8
1040 PMBASE=INT(PMBASE/DIVISOR-1)*DIVI
SOR
```

Speichern der Player und Missiles in der Tabelle

Nachdem Sie die Anfangsadresse der Tabelle festgelegt haben, können Sie nun die Player und Missiles in der Tabelle speichern. Zunächst müssen Sie die Abschnitte der Tabelle löschen, die Sie zur Speicherung verwenden wollen. Der erste Abschnitt der Tabelle wird nicht benutzt. Es ist nicht notwendig, diesen oder einen anderen Abschnitt der Tabelle, den Sie nicht benötigen, zu löschen.

Steuerung der Player-Missile-Grafik

Es gibt mehrere Steuerregister, mit deren Hilfe, wie der Name schon sagt, die Player-Missile-Grafik gesteuert werden kann:

- Ein Register für die Anfangsadresse der Tabelle
- Jeweils ein Steuerregister für DMA und die Grafikdarstellung
- Register für die Festlegung der Breite der Player und Missiles
- Farbregister für Player und Missiles
- Register für die Festlegung der horizontalen Position der Player und Missiles
- Prioritätsregister

Einigen Registern muß nur einmal am Anfang des Programmes ein bestimmter Wert zugewiesen werden. Auf andere Register muß eventuell ständig zugegriffen werden, je nachdem, wie Ihr Programm aufgebaut ist und wie die Grafikdarstellung aussehen soll. In den ATARI-Manuals werden für diese Register Abkürzungen verwendet. Diese Abkürzungen finden Sie auch in den Überschriften der nun folgenden Abschnitte.

Das Player-Missile Base Register (PMBASE)

In den Speicherzellen 54279 und 54280 (D407 und D408 hexadezimal) wird die Anfangsadresse der Tabelle zur Darstellung der Player-Missile-Grafiken (= Player-Missile Base) gespeichert. Da es sich dabei immer um eine 1K- oder 2K-Grenze handeln muß, muß in der Speicherzelle 54279 immer eine 0 abgelegt werden. Nur das höherwertige Byte der Adresse (die Nummer der Page) spielt hier eine Rolle.

Das Graphics Control Register (GRACTL)

Zusammen mit dem im nächsten Abschnitt beschriebenen DMA-Steuerregister wird mit Hilfe dieses Registers der direkte Speicherzugriff (DMA) zur Darstellung der Player-Missile-Grafiken aktiviert. Auf dieses Steuerregister (=Control Register) kann über die Speicherzelle 53277 (D01D hexadezimal) zugegriffen werden. Sie können wählen, ob nur die Player (POKE 53277,2), nur die Missiles (POKE 53277,1) oder beides (POKE 53277,3) dargestellt werden soll.

Das DMA Control Register (DMACTL)

Mit Hilfe dieses Registers wird die Player-Missile-Grafik ein- und ausgeschaltet. Allerdings muß gleichzeitig auch GRACTL aktiviert sein. Anderenfalls würden Sie trotzdem nur die Playfield-Grafik sehen. DMACTL und GRACTL müssen also zusammen benutzt werden. Auch bei Benutzung von DMA wird der 6502-Mikroprozessor zeitlich etwas belastet, weil der 6502 und ANTIC nicht gleichzeitig auf den Speicher zugreifen können. Wenn Sie keine Player-Missile-Grafik mehr darstellen wollen, sollten Sie DMACTL rücksetzen, wodurch sich die Rechengeschwindigkeit etwas erhöht.
Auf DMACTL können Sie über die Speicherzelle 559 (22F hexadezimal) zugreifen. In Tabelle 9-3 sind nebeneinander die Werte, die Sie dort mit Hilfe eines POKE-Befehls ablegen können und die entsprechende Betriebsart aufgeführt.

Tabelle 9-3: Funktionen des DMA-Steuerregisters

Wert des Registers	Betriebsart
4	Darstellung der Missiles
8	Darstellung der Player
12	Darstellung der Player und der Missiles
+16	Für doppelte Auflösung (die angegebenen Werte gelten für einfache Auflösung)

Die Register zur Festlegung der Breite der Player (SIZEP0-SIZEP3)

Mit Hilfe von vier 8-Bit-Registern, auf die über die Speicherzellen 53256 bis 53259 (D008 bis D00B hexadezimal) zugegriffen werden kann, wird die Breite der Player festgelegt. Durch Ablegen bestimmter Zahlen in diesen Speicherzellen können Sie die Breite (nicht die Höhe) eines Players verdoppeln oder vervierfachen. Wenn Sie den Inhalt dieser Register nicht verändern, werden die Player in normaler Größe dargestellt. Speicherzelle 53256 bestimmt die Breite des ersten Players, Speicherzelle 53257 die des zweiten usw.. Wenn Sie in einem Programm einen Player in horizontaler Richtung über den Bildschirm bewegen, dann wird die Position des Players durch diese Register beeinflußt (siehe auch den in diesem Kapitel enthaltenen Abschnitt über die Register zur Festlegung der horizontalen Position der Player). Wenn ein Player in doppelter Breite dargestellt werden soll, legen Sie in der entsprechenden Speicherzelle eine 1, wenn er in vierfacher Breite dargestellt werden soll, eine 3 ab. Falls der Wert eines Registers 0 oder 2 beträgt, wird der dazugehörige Player in normaler Größe dargestellt. Mit Hilfe des Befehls POKE 53258,1 wird zum Beispiel die Breite des dritten Players verdoppelt.

Die Register zur Festlegung der Breite des Missiles (SIZEM)

Über die Speicherzelle 53260 (D00C hexadezimal) kann auf ein Register zugegriffen werden, mit dessen Hilfe die Breite der Missiles festgelegt wird. Die Breite aller Missiles ist gleich. Die Funktion dieses Registers entspricht im übrigen der Funktion der Register für die Festlegung der Breite der Player: eine 0 oder 2 bedeutet normale Breite, eine 1 doppelte Breite und eine 3 vierfache Breite.

Die Farbregister für Player und Missiles (COLPM0-COLPM3)

Über die Speicherzellen 704 bis 707 (2C0 bis 2C3 hexadezimal) kann auf vier 8-Bit-

Tabelle 9-4: Inhalt der Farbregister zur Erzeugung einer bestimmten Farbe und Helligkeit (gilt auch für die Playfield-Grafiken).

	Inhalt des Farbregisters*	
	dezimal	hexadezimal
Grau	0	0
Gold	16	10
Orange	32	20
Rot	48	30
Rosa	64	40
Violett	80	50
Purpur	96	60
Blau	112	70
Blau	128	80
Hellblau	144	90
Türkis	160	A0
Blau-Grün	176	B0
Grün	192	C0
Gelb-Grün	208	D0
Orange-Grün	224	E0
Hellorange	240	F0

* Zur Festlegung der Helligkeit addieren Sie zu den angegebenen Werten eine gerade Zahl zwischen 0 und 14 (0 = dunkel, 14 = maximale Helligkeit).

Farbregister zugegriffen werden. Zueinander gehörende Player und Missiles bekommen dieselbe Farbe. Speicherzelle 704 bestimmt die Farbe des ersten Player-Missile-Paares, Speicherzelle 705 die des zweiten usw.. Tabelle 9-4 können Sie die Werte entnehmen, die Sie in diesen Registern zur Erzeugung einer bestimmten Farbe und Helligkeit ablegen müssen.

Die Register zur Festlegung der horizontalen Position der Player (HPOSP0-HPOSP3)

Um einen Player von einer Stelle des Bildschirms an eine andere zu bewegen (in horizontaler Richtung), brauchen Sie nur den Inhalt des entsprechenden Registers zur Festlegung der horizontalen Position dieses Players mit Hilfe eines POKE-Befehls zu verändern. Wenn sich ein Player zum Beispiel an der linken Bildschirmseite befindet und Sie den Inhalt des zu diesem Player gehörenden Registers verändern, dann wird der Player sofort an der alten Position verschwinden und praktisch im selben Moment an der neuen Position auftauchen. Der Wert der Register darf zwischen 0 und 227 liegen. Dieser Bereich kann sich jedoch, je nachdem, in welcher Größe die Player dargestellt werden, reduzieren auf Werte zwischen 40 (linke Begrenzung) und 190 (rechte Begrenzung), wenn die Player vollständig sichtbar sein sollen.
Diese Register können nicht gelesen werden, das heißt, es ist nicht möglich, mit Hilfe eines PEEK-Befehls die Position eines Players festzustellen. Zu diesem Zweck müssen in Ihrem Programm Variablen vorhanden sein. Ein Beispiel dafür, wie diese Register zu benutzen sind, folgt später. Die Speicherzelle 53248 (D000 hexadezimal) ist für die horizontale Position des ersten Players zuständig, Speicherzelle 53249 für die des zweiten, Speicherzelle 53250 für die des dritten und Speicherzelle 53251 für die des vierten.

Die Register zur Festlegung der horizontalen Position der Missiles (HPOSM0-HPOSM3)

Ab der Speicherzelle 53252 (D004 hexadezimal) kann auf vier Register zur Festlegung der horizontalen Position der Missiles zugegriffen werden.

Beispiele zur Benutzung der Player-Missile-Grafik

In diesem Kapitel werden Sie einige Tricks kennenlernen, die bei Benutzung der Player-Missile-Grafik sehr von Nutzen sind. In allen hier vorgestellten Programmen werden Unterprogramme zum Erstellen, Initialisieren und Laden der Tabelle für die Darstellung der Player-Missile-Grafiken verwendet, die Sie bereits kennengelernt haben. Im Verlauf des vorigen Kapitels haben Sie jedoch nicht erfahren, wie Player oder Missiles in vertikaler Richtung, also nach oben oder unten, bewegt werden können. Auf dieses Problem wird im Folgenden näher eingegangen werden.

Bewegung von Playern und Missiles in vertikaler Richtung

Ein Player oder Missile kann vertikal bewegt werden, indem die zur Festlegung seiner Form notwendige Information in der Tabelle für die Darstelllung der Player-Missile-Grafiken nach oben (um das Objekt auf dem Bildschirm nach unten zu bewegen) oder nach unten (um das Objekt auf dem Bildschirm nach oben zu bewegen) verschoben wird, selbstverständlich immer innerhalb des Abschnittes der Tabelle, der für dieses Objekt zuständig ist. Im BASIC wäre dieser Vorgang viel zu langsam. Aus diesem Grund verwendet man hier gerne ein Assemblerprogramm. Mit Hilfe des folgenden Programmes kann die oben beschriebene Verschiebung durchgeführt werden:

```
0000      0010          *=$0600          ;ANFANGSADRESSE DES PROGRAMMES
00CB      0020 ADRESSE  =$00CB           ;ANFANGSADRESSE DES PLAYERS
00CD      0030 LAENGE   =$00CD           ;LAENGE DES PLAYERS (IN BYTES)
0600 68   0040 OBEN     PLA              ;VOM STACK LADEN (SIEHE USR)
0601 A001 0050          LDY #$01         ;INDEXREGISTER INITIALISIEREN
0603 B1CB 0060 WEITOBEN LDA (ADRESSE),Y
0605 88   0070          DEY              ;UM EIN BYTE NACH OBEN VERSCHIEBEN
0606 91CB 0080          STA (ADRESSE),Y
0608 C4CD 0090          CPY LAENGE       ;VERSCHIEBUNG BEENDET?
060A C8   0100          INY              ;REGISTER UM 2 ERHOEHEN
060B C8   0110          INY
060C 90F5 0120          BCC WEITOBEN     ;NICHT FERTIG > WEITERVERSCHIEBEN
060E C6CB 0130          DEC ADRESSE
0610 60   0140          RTS              ;FERTIG > RETURN
0611 68   0150 UNTEN    PLA              ;AUCH HIER VOM STACK LADEN
0612 A4CD 0160          LDY LAENGE
0614 B1CB 0170 WEITUNTEN LDA (ADRESSE),Y
0616 C8   0180          INY              ;UM EIN BYTE NACH UNTEN VERSCHIEBEN
0617 91CB 0190          STA (ADRESSE),Y
0619 88   0200          DEY              ;REGISTER UM 2 ERNIEDRIGEN
061A 88   0210          DEY
061B 10F7 0220          BPL WEITUNTEN    ;NICHT FERTIG > WEITERVERSCHIEBEN
061D E6CB 0230          INC ADRESSE
061F 60   0240          RTS
```

Dieses Unterprogramm können Sie mit Hilfe des folgenden Programmes beziehungsweise Programmteiles in ein BASIC-Programm einfügen:

```
4000 REM ****************************
4001 REM *UNTERPROGRAMM ZUR BEWEGUNG*
4002 REM *  EINES PLAYERS/MISSILES  *
4003 REM *==========================*
4004 REM *BEI DER AUSFUEHRUNG DIESES*
4005 REM *   PROGRAMMES WIRD DAS    *
4006 REM * UNTERPROGRAMM AB ADRESSE *
4007 REM *1536 IM SPEICHER ABGELEGT,*
4008 REM ****************************
4010 DATA 104,160,1,177,203,136,145
4020 DATA 203,196,205,200,200,144,245
4030 DATA 198,203,96,104,164,205,177
4040 DATA 203,200,145,203,136,136
4050 DATA 16,247,230,203,96
4060 FOR I=1536 TO 1567
4070 READ J
4080 POKR I,J
4090 NEXT I
```

Der Speicherbereich von 1536 bis 1791 wird vom ATARI-Computer nicht benutzt. Wenn Sie ein Assemblerprogramm dort speichern, können Sie sicher sein, daß es nicht überschrieben wird. Es wird Ihnen so lange zur Verfügung stehen, bis Sie den Computer ausschalten oder diesen Speicherbereich anderweitig nutzen. Die nun folgenden Beispiele zur Benutzung der Player-Missile-Grafik funktionieren nur im Zusammenhang mit dem oben aufgeführten Unterprogramm. Vor der Ausführung eines Beispielprogrammes muß folglich dieses Unterprogramm geladen werden.

Ein einfaches Programm zur Bewegung eines Players

In diesem Programm wird nur ein Player benutzt und dieser Player wird nur in horizontaler Richtung bewegt. Mit Hilfe des DATA-Befehls in Zeile 30 wird die Form des Players festgelegt. Nachdem der für diesen Player zuständige Abschnitt der Tabelle für die Darstellung der Player-Missile-Grafiken gelöscht worden ist, wird er in diesem Abschnitt gespeichert. Dann werden die Steuerregister (PMBASE, GRACTL und DMACTL) initialisiert. Die Darstellung des Players erfolgt in einfacher Auflösung und normaler Größe. Die Farbe und Helligkeit des Players (in diesem Fall blau) wird mit Hilfe des POKE-Befehls in Zeile 130 festgelegt. Dann wird der Inhalt des Registers zur Festlegung der horizontalen Position des Players schrittweise erhöht und damit der Player von links nach rechts über den Bildschirm bewegt (Zeile 190 bis 210). Nun können Sie mit diesem Programm ein wenig experimentieren und sich so mit der Funktionsweise der Player-Missile-Grafik vertraut machen. Verändern Sie zum Beispiel einmal die Breite des Players oder den Inhalt des Farbregisters oder ändern Sie die FOR-NEXT-Schleife so, daß sich der Player nicht von links nach rechts, sondern von rechts nach links über den Bildschirm bewegt.

```
10 GRAPHICS 0
20 SETCOLOR 2,0,0
25 REM FESTLEGUNG DER FORM DES PLAYERS
 MIT HILFE DES FOLGENDEN DATA-BEFEHLS
30 DATA 24,60,255,36,66
40 Y=64:REM FESTLEGUNG DER VERTIKALEN
POSITION DES PLAYERS
50 A=PEEK(106)-8:REM ENDE DES SPEICHER
S SUCHEN
60 POKE 54279,A:REM ANFANGSADRESSE DER
 TABELLE IN PMBASE ABLEGEN
70 ANF=256*A+512:REM ANFANGSADRESSE DE
S ABSCHNITTS FUER PLAYER 0
80 POKE 559,64:REM DMACTL INITIALISIER
EN
90 POKE 53277,3:REM GRACTL INITIALISIE
REN
100 FOR I=ANF TO ANF+127
110 POKE I,0:REM LOESCHEN DES FUER PLA
YER 0 ZUSTAENDIGEN ABSCHNITTS DER TABE
LLE
120 NEXT I
130 POKE 704,136:REM FARBE VON PLAYER
0 FESTLEGEN
140 POKE 53248,0:REM FESTLEGUNG DER HO
RIZONTALEN POSITION VON PLAYER 0
150 FOR I=ANF+Y TO ANF+4+Y
160 READ X:REM INFORMATION ZUR DARSTEL
LUNG DES PLAYERS IN DER TABELLE ABLEGE
N
170 POKE I,X
180 NEXT I
190 FOR POS=0 TO 228
200 POKE 53248,POS:REM PLAYER UM EINEN
 FARBZYKLUS NACH RECHTS BEWEGEN
210 NEXT POS
220 GOTO 190
```

Programm zur horizontalen und vertikalen Bewegung eines Players

Das in Bild 9-10 aufgeführte Programm ist wesentlich aufwendiger als das vorhergehende. Hier wird das Assemblerprogramm zur vertikalen Bewegung des Players benutzt. Sie werden feststellen, daß auch die FOR-NEXT-Schleifen ein wenig anders aufgebaut sind. Auf diese Weise kann ein dreidimensionaler Eindruck erzeugt werden.

Darstellung eines Players mit höherer Auflösung

Für manche Anwendungen können Sie die Player wegen ihrer zu geringen Auflösung nicht verwenden. Dieses Problem kann jedoch umgangen werden, wenn Sie zur Darstellung eines Objektes mehrere Player verwenden. So wurde auch in dem in Bild 9-11 aufgeführten Programm verfahren. Das mit Hilfe dieses Programmes dargestellte Objekt mit 32 Bit Breite finden Sie in Bild 9-12.
Die Bewegung dieses Objektes auf dem Bildschirm erfolgt allerdings etwas ruckartig. Der Grund dafür ist darin zu suchen, daß das BASIC für die Ausführung dieser Funktion zu langsam ist.

Benutzung der Prioritätsregister

Ein Programmbeispiel für die Benutzung der Prioritätsregister finden Sie in Bild 9-13. Das Prioritätsregister kann die folgenden Werte annehmen:

1 : alle Player haben Priorität gegenüber dem Playfield
2 : Player 0 und 1 haben Priorität gegenüber dem Playfield und den Playern 2 und 3
4 : Playfield hat Priorität gegenüber den Playern
8 : die Farbregister 0 und 1 des Playfields haben Priorität gegenüber den Playern und den Farbregistern 2 und 3 des Playfields

In unserem Beispiel scheint sich der Player vor dem gelben Teil des Bildschirms zu befinden. Bevor der Player an die linke Bildschirmseite bewegt wird, wird der Inhalt des Prioritätsregisters geändert. Jetzt hat das gelbe Playfield Priorität gegenüber dem Player. Das Ganze sieht dann so aus, als wenn der Player hinter dem gelben Playfield verschwindet.

```
10 GRAPHICS 1+16
20 SETCOLOR 0,0,12
30 SETCOLOR 2,0,0
40 DATA 153,189,255,189,153
50 GOSUB 360:REM ZEICHNEN EINES STERNE
NHINTERGRUNDES
60 Y=1
70 A=PEEK(106)-8
80 POKE 106,A
90 POKE 54279,A:REM ANFANGSADRESSE DER
 TABELLE IN PMBASE ABLEGEN
100 ANF=256*A
110 POKE 559,62:REM DOPPELTE AUFLOESUN
G
```

Bild 9-10: Bewegung eines Players in horizontaler und vertikaler Richtung

```
120 POKE 53277,3:REM DMA AKTIVIEREN
130 POKE 53256,0
140 FOR I=ANF+1024 TO ANF+1280
150 POKE I,0:REM LOESCHEN DER TABELLE
160 NEXT I
170 PANF=ANF+1024+Y
180 POKE 204,INT(PANF/256):REM HOEHERW
ERTIGES BYTE DER ANFANGSADRESSE DES PL
AYERS ABLEGEN
190 POKE 203,PANF-(PEEK(204)*256)-1:RE
M NIEDERWERTIGES BYTE DER ANFANGSADRES
SE DES PLAYERS ABLEGEN
200 POKE 205,5:REM GROESSE DES PLAYERS
210 RESTORE
220 FOR I=PANF TO PANF+4
230 READ A
240 POKE I,A
250 NEXT I
260 FOR I=50 TO 120:GOSUB 320:NEXT I
270 POKE 53256,1:REM BREITE DES PLAYER
S VERDOPPELN
280 FOR I=118 TO 167 STEP 2:GOSUB 320:
GOSUB 320:NEXT I
290 POKE 53256,3:REM BREITE DES PLAYER
S VERVIERFACHEN
300 FOR I=166 TO 250 STEP 2:GOSUB 320:
NEXT I
310 GOTO 130
320 A=USR(1553):POKE 53248,I:FARBE=FAR
BE+1
330 IF FARBE>255 THEN FARBE=0
340 POKE 704,FARBE
350 RETURN
360 FOR M=1 TO 22
370 X=RND(0)*20
380 Y=RND(1)*24
390 POSITION INT(X),INT(Y):? #6;".":RE
M ZEICHNEN DER STERNE MIT HILFE DES ZU
FALLSGENERATORS
400 X=RND(0)*20
410 Y=RND(1)*24
420 POSITION INT(X),INT(Y):PRINT #6;".
"
430 NEXT M
440 RETURN
```

Bild 9-10: Bewegung eines Players in horizontaler und vertikaler Richtung (Fortsetzung)

```
1 REM PLAYER MIT 32 BIT BREITE
2 REM ES FOLGEN DIE DATEN ZUR DARSTELL
UNG DES PLAYERS
3 SETCOLOR 2,0,0
10 DATA 0,0,0,0,3,15,119,254,255,63,31
,7,4,14,14
20 DATA 1,2,2,31,240,255,255,255,239,2
54,255,255,240,15,0
30 DATA 128,64,64,248,15,255,255,255,2
47,127,255,255,15,240,0
40 Y=80
45 DATA 0,0,0,0,192,240,236,127,255,25
2,248,224,32,112
50 A=PEEK(106)-8:REM ENDE DES SPEICHER
S BESTIMMEN
60 POKE 54279,A:REM ANFANGSADRESSE DER
 TABELLE IN PMBASE ABLEGEN
70 ANF=256*A:REM ANFANGSADRESSE DER TA
BELLE ZUR DARSTELLUNG DER PLAYER-MISSI
LE-GRAFIK
80 POKE 559,46:REM DMACTL INITIALISIER
EN
90 POKE 53277,3:REM GRACTL INITIALISIE
REN
100 FOR I=ANF+512 TO ANF+1024
110 POKE I,0
120 NEXT I
130 FOR I=704 TO 708
140 POKE I,188
150 NEXT I:REM FARBE DES PLAYERS=AQUA
160 FOR H=0 TO 3
170 POKE 53248+H,X+H*8:REM FESTLEGUNG
DER HORIZONTALEN POSITION
180 POKE 53256+H,1:REM FESTLEGUNG DER
GROESSE
190 FOR I=ANF+512+Y+(128*H) TO ANF+526
+Y+(128*H)
200 READ X
210 POKE I,X
220 NEXT I
225 NEXT H
230 FOR D=0 TO 228 STEP 4
240 FOR P=0 TO 3
250 POS=D+16*P
260 IF POS<228 THEN POKE 53248+P,POS
270 NEXT P
280 NEXT D
290 GOTO 230
```

Bild 9-11: Darstellung eines Players mit einer Breite von 32 Bit

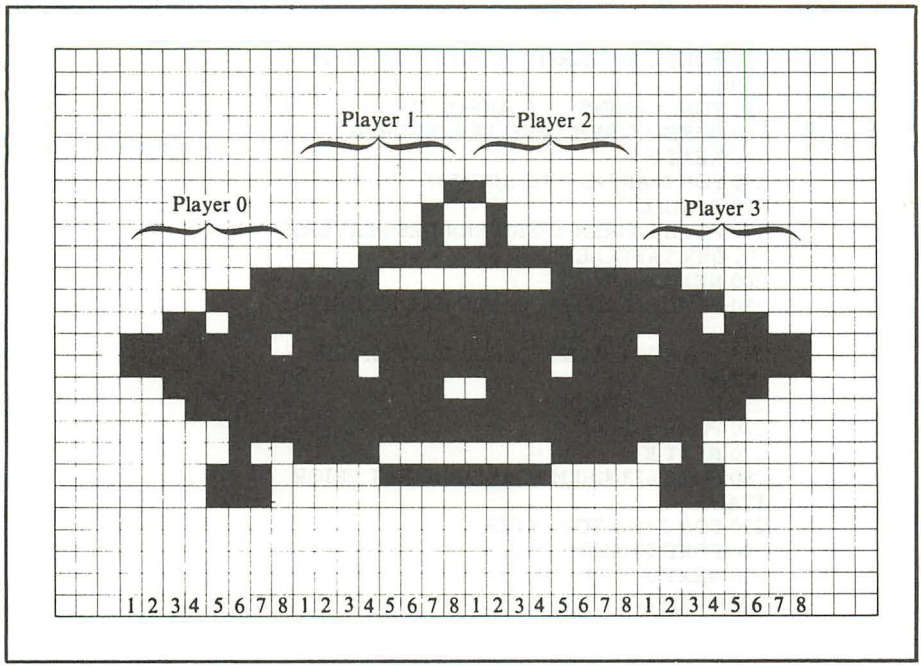


Bild 9-12: Zusammenfügen mehrerer Player

```
5 COLOR 3
10 GRAPHICS 7+16
20 DATA 60,126,219,255,189,195,126,60
30 SETCOLOR 2,0,0
40 PLOT 125,95:REM ZEICHNEN EINES OBJE
KTES AUF DEM PLAYFIELD
50 DRAWTO 80,0
60 DRAWTO 79,0
70 POSITION 45,95
80 POKE 765,1:REM ORANGEFARBENES DREIE
CK
85 XIO 18,#6,0,0,"S:"
90 A=PEEK(106)-24:REM ENDE DES SPEICHE
RS BESTIMMEN
100 POKE 54279,A:REM ANFANGSADRESSE DE
R TABELLE IN PMBASE ABLEGEN
110 ANF=256*A:REM ANFANGSADRESSE DER T
ABELLE ZUR DARSTELLUNG DER PLAYER-MISS
ILE-GRAFIK
120 POKE 559,46:REM DMACTL INITIALISIE
REN
```

Bild 9-13: Benutzung der Prioritätsregister

```
130 POKE 53277,3:REM GRACTL INITIALISI
EREN
140 FOR J=ANF+512 TO ANF+1024
150 POKE J,0:REM TABELLE LOESCHEN
160 NEXT J
170 FOR I=ANF+580 TO ANF+587
180 READ X
190 POKE I,X:REM SPEICHERN DES PLAYERS
 IN DER TABELLE
200 NEXT I
230 POKE 623,1:REM PLAYER HAT PRIORITA
ET
240 POKE 704,86:REM FARBE DES PLAYERS=
PURPUR
250 FOR K=60 TO 180
260 POKE 53248,K
270 NEXT K
280 POKE 623,4:REM PLAYFIELD HAT PRIOR
ITAET
290 FOR J=180 TO 60 STEP -1
300 POKE 53248,J
310 NEXT J
320 GOTO 230
```

Bild 9-13: Benutzung der Prioritätsregister (Fortsetzung)

10

TONERZEUGUNG MIT DEM ATARI-COMPUTER

Töne können beim ATARI-Computer auf zwei verschiedene Arten erzeugt werden: erstens mit Hilfe des eingebauten Lautsprechers und zweitens mit Hilfe des Lautsprechers des angeschlossenen Fernsehgerätes.

Der eingebaute Lautsprecher

Jedesmal, wenn Sie eine Taste drücken, wird der eingebaute Lautsprecher kurz aktiviert. Außerdem wird er bei der Bedienung des Programmrecorders benutzt. Über die Speicherzelle 53279 können Sie auf ihn zugreifen. Wenn Sie dort eine 0 ablegen, wird als Folge davon ein Impuls erzeugt, mit dem der Lautsprecher angesteuert wird. Wenn die Impulse schnell aufeinander folgen, ergibt das einen Ton. Je schneller die Impulse aufeinander folgen, desto höher ist dieser Ton. Das folgende Programm veranschaulicht diesen Sachverhalt:

```
10 PRINT "TONHOEHE (1=HOCH, 10=TIEF)";
20 INPUT T
25 REM DIESE SCHLEIFE BESTIMMT DIE TON
DAUER
30 FOR J=1 TO 15
40 POKE 53279,0:REM LAUTSPRECHER
45 REM DIESE SCHLEIFE BESTIMMT DIE TON
HOEHE
50 FOR K=1 TO T
60 NEXT K
70 NEXT J
80 GOTO 10
```

Hohe Töne können auf diese Art nicht erzeugt werden, weil das BASIC dafür nicht schnell genug ist. Trotzdem gibt es einige Anwendungen, bei denen sich der eingebaute Lautsprecher als sehr nützlich erweisen wird. Sie könnten zum Beispiel das Unterprogramm zur Ausgabe einer Fehlermeldung (Bild 4-17) so modifizieren, daß gleichzeitig mit der Fehlermeldung der eingebaute Lautsprecher einen Ton erzeugt.

Tonerzeugung mit Hilfe des Lautsprechers des Fernsehgerätes

In Verbindung mit dem Lautsprecher des Fernsehgerätes kann der ATARI-Computer eine Fülle von Tönen erzeugen. Dabei kann ein-, zwei-, drei- oder vierstimmig gearbeitet werden. Jede Stimme umfaßt mehr als drei Oktaven. Es können sowohl reine als auch stark verzerrte Töne erzeugt werden. Unabhängig von der Einstellung des Lautstärkerreglers am Fernsehgerät kann die Lautstärke für jede Stimme einzeln festgelegt werden.

Der SOUND-Befehl

Mit Hilfe des SOUND-Befehls können Sie vom BASIC aus den Lautsprecher des Fernsehgerätes ansprechen. Drehen Sie den Lautstärkeregler Ihres Fernsehgerätes ein wenig auf und versuchen Sie einmal folgendes Beispiel:

```
SOUND 0,121,10,8
```

Bei dem Ton, den Sie daraufhin hören, handelt es sich um das mittlere C. Jeder SOUND-Befehl muß von vier Zahlen gefolgt sein (Bild 10-1). Mit Hilfe der ersten Zahl wird festgelegt, welche Stimme den Ton erzeugen soll. Die zweite Zahl bestimmt die Tonhöhe, die dritte die Reinheit des Tones und die vierte die Lautstärke. An Stelle von Konstanten können Sie genausogut numerische Variablen oder arithmetische Ausdrücke verwenden.
Ein einmal eingeschalteter Ton ertönt so lange, bis Sie ihn wieder abschalten. Dazu müssen Sie nur die Tonhöhe, die Reinheit des Tones und die Lautstärke auf 0 setzen, zum Beispiel so:

```
SOUND 0,0,0,0
```

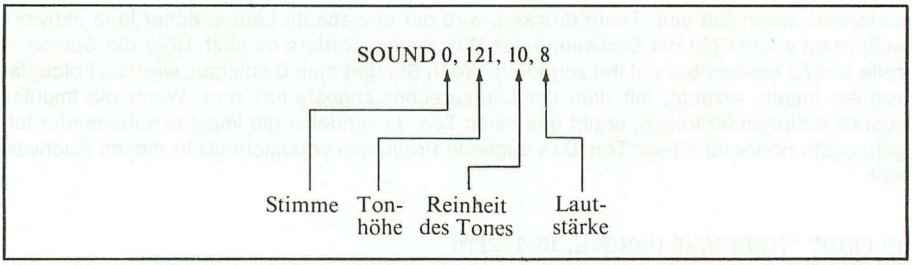


Bild 10-1: Die verschiedenen Parameter des SOUND-Befehls

Die Stimmen

Beim ATARI-Computer gibt es vier verschiedene Stimmen, die vollkommen unabhängig voneinander sind. Es können also vier verschiedene Töne gleichzeitig erzeugt werden. Die Stimmen überlagern sich im Lautsprecher des Fernsehgerätes, genauso wie die verschiedenen Stimmen eines Chors. Mit Hilfe der ersten auf den SOUND-Befehl folgenden Zahl

wird festgelegt, auf welche Stimme sich der SOUND-Befehl bezieht. Die verschiedenen Stimmen sind mit den Zahlen von 0 bis 3 gekennzeichnet. Jede Stimme muß mit einem eigenen SOUND-Befehl angesprochen werden. Die unten aufgeführte Befehlsfolge erzeugt zum Beispiel einen C-Akkord:

```
SOUND 0,121,10,8

READY
SOUND 1,96,10,8

READY
SOUND 2,81,10,8

READY
SOUND 3,60,10,8

READY
```

Mit Hilfe der folgenden FOR-NEXT-Schleife können Sie einen Ton, den Sie vorher eingeschaltet haben, wieder abschalten:

```
FOR J=0 TO 3:SOUND J,0,0,0:NEXT J
```

Die Tonhöhe

Die zweite auf einen SOUND-Befehl folgende Zahl bestimmt die Tonhöhe. Diese Zahl muß zwischen 0 und 255 liegen. Der ATARI-Computer kann alle Noten (auch Halbtöne) von einer Oktave unter dem mittleren C bis zwei Oktaven darüber erzeugen (Bild 10-2). Zusätzlich können noch eine ganze Menge anderer Töne erzeugt werden. Zwischen dem mittleren C und der einen Halbton darunter befindlichen Note (B) gibt es zum Beispiel noch sechs Zwischenwerte. Da diese Töne in der Tonleiter nicht vorkommen, wird man sie bei der Programmierung von Musik nicht verwenden. Sie können allerdings sehr wohl für andere Effekte verwendet werden. Probieren Sie einmal das folgende Programm aus:

```
10 FOR J=-255 TO 255
20 SOUND 0,ABS(J),10,8
30 PRINT "TONHOEHE: ";ABS(J)
40 FOR K=1 TO 50:NEXT K
50 NEXT J
60 SOUND 0,0,0,0
```

Alle Töne, die der ATARI-Computer erzeugen kann, werden in diesem Programm verwendet. Wenn Sie sich das Ergebnis dieses Programmes anhören, werden Sie feststellen, daß die Tondauer der tiefen Töne länger zu sein scheint als die der hohen Töne. Dem Programmlisting können Sie aber entnehmen, daß die Tondauer für jeden Ton exakt gleich lang ist (Zeile 40). Wenn Sie jedoch für die Tonhöhe 255 einsetzen, dann gibt es keinen großen Unterschied zu den Tönen, die erzeugt werden, wenn für die Tonhöhe 254, 253 oder sogar 252 eingesetzt wird. Diese tiefen Töne gehen ineinander über und klingen dann wie ein einziger langanhaltender Ton. Demgegenüber werden Sie jedoch sehr wohl einen Unterschied zwischen zwei Tönen feststellen, wenn Sie bei einem für die Tonhöhe 11 und bei dem anderen 10 einsetzen. Hier ist jede Änderung der Tonhöhe sofort zu hören. Zusammenfassend kann man also sagen, daß die tiefen Töne ineinander übergehen, während die hohen Töne sehr gut voneinander unterschieden werden können. Daraus ergibt sich demnach der oben beschriebene Effekt, daß die tiefen Töne länger zu dauern scheinen als die hohen.

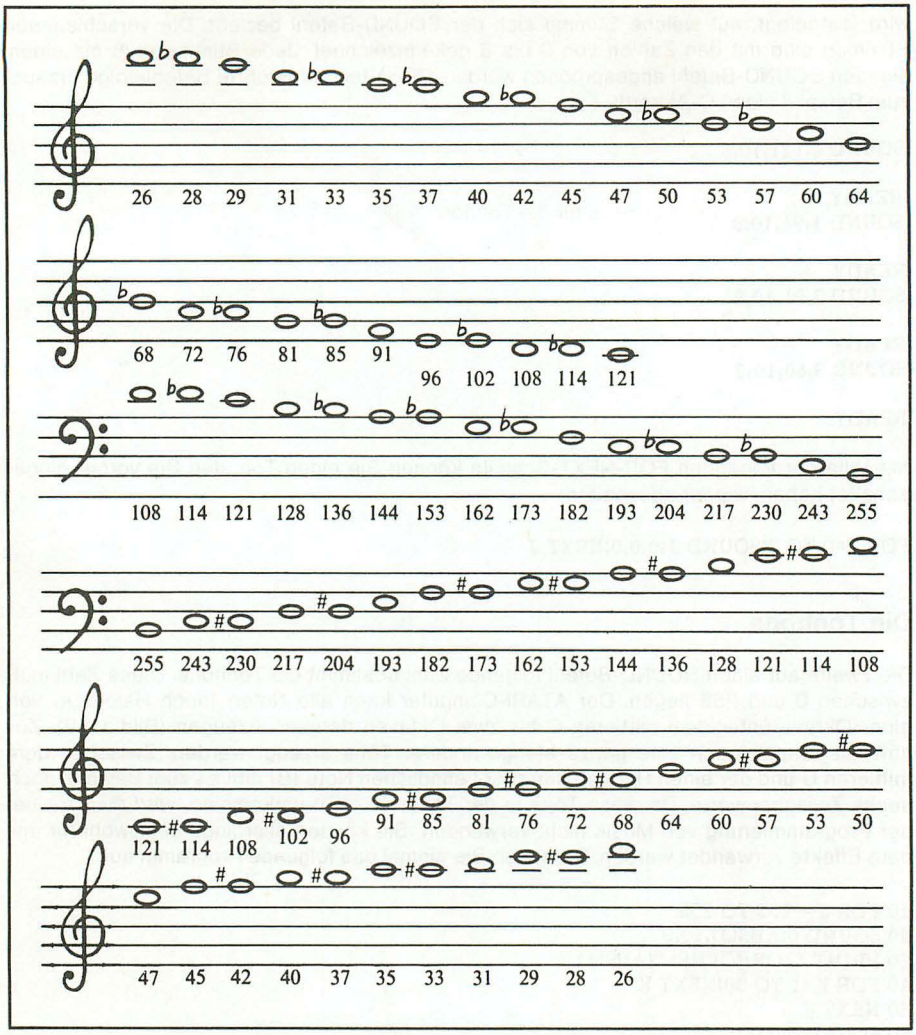


Bild 10-2: Die Tonleiter mit den für die verschiedenen Töne einzusetzenden Wert für die Tonhöhe

Die Reinheit der erzeugten Töne

Mit dem ATARI-Computer können sowohl reine als auch verzerrte Töne erzeugt werden. Die dritte auf einen SOUND-Befehl folgende Zahl ist hierfür zuständig. Diese Zahl muß zwischen 0 und 15 liegen. Bei Angabe einer 10 oder 14 wird ein reiner Ton erzeugt. Bei Angabe einer anderen geraden Zahl (0, 2, 4, 6, 8, 10 oder 12) werden die Töne mehr oder weniger stark verzerrt. Die Stärke der Verzerrung hängt sowohl von der für die Reinheit des Tones angegebenen Zahl als auch von der Tonhöhe des Tones ab.
Bei bestimmten Kombinationen von Verzerrung und Tonhöhe wird ein unverzerrter Ton mit harmonischen Übertönen erzeugt. Die Tonhöhe dieses Tones unterscheidet sich von der des entsprechenden reinen Tones. Ein Beispiel soll das veranschaulichen. Erzeugen Sie zunächst mit Hilfe des folgenden Befehls ein reines C#:

```
SOUND 0,230,10,8
```

Verändern Sie nun die Reinheit des Tones, indem Sie die 10 durch eine 12 ersetzen:

```
SOUND 0,230,12,8
```

Sie hören nun einen sehr viel tieferen Ton. Dieser Ton ist sogar noch tiefer als der reine Ton, der erzeugt wird, wenn Sie für die Tonhöhe 255 einsetzen:

```
SOUND 0,255,10,8
```

Leider ist die Tonhöhe dieser Töne nicht exakt festgelegt. Das folgende Programm veranschaulicht das:

```
5 REM ERZEUGUNG VON UNVERZERRTEN TOENE
N BEI BESTIMMTEN KOMBINATIONEN VON VER
ZERRUNG UND TONHOEHE
10 SOUND 0,230,12,8
20 FOR K=1 TO 50:NEXT K
25 REM ABSCHALTEN DES TONES
30 SOUND 0,0,0,0
35 REM EINFUEGEN EINER ZEITSCHLEIFE, D
EREN LAENGE MIT HILFE DES ZUFALLSGENER
ATORS FESTGELEGT WIRD
40 FOR K=1 TO 50*RND(0):NEXT K
45 WIEDERHOLUNG DES GANZEN VORGANGES;
BEENDEN DES PROGRAMMES MIT BREAK
50 GOTO 10
```

Die Zeit, die vom Abschalten des Tones bis zum Wiedereinschalten vergeht, ist unterschiedlich lang (Zeile 40). Obwohl der SOUND-Befehl immer derselbe ist (Zeile 10), wird dadurch die Tonhöhe beeinflußt.
Einige Kombinationen von Verzerrung und Tonhöhe löschen sich gegenseitig aus. Der folgende Befehl ist ein Beispiel dafür. Bei seiner Ausführung werden Sie gar nichts hören:

```
SOUND 0,29,4,8
```

Wenn Sie bei einem SOUND-Befehl für die Verzerrung eine ungerade Zahl angeben (1, 3, 5 und so weiter), wird die mit diesem Befehl angesprochene Stimme abgeschaltet. Wenn die Stimme jedoch bereits abgeschaltet war, wird ein kurzer Impuls erzeugt. Danach werden Sie nichts mehr hören, obwohl die Stimme ja immer noch eingeschaltet ist. Wenn Sie sie nun wieder abschalten, wird noch einmal ein kurzer Impuls erzeugt. Das folgende Programm veranschaulicht diesen Sachverhalt:

```
10 FOR J=1 TO 10
20 SOUND 0,0,1,8
25 FOR K=1 TO 100:NEXT K
30 SOUND 0,0,0,0
40 FOR K=1 TO 100:NEXT K
50 NEXT J
```

In Tabelle 10-1 finden Sie eine Zusammenstellung aller möglichen Werte, die Sie für die Verzerrung angeben können, und der charakteristischen Eigenschaften der damit erzeugten Töne. Das folgende Programm soll die Auswirkungen der verschiedenen Verzerrungen aufzeigen:

```
10 FOR P=0 TO 255
20 FOR D=0 TO 15
30 PRINT "HOEHE=";P,"VERZERRUNG=";D
40 SOUND 0,P,D,10
50 FOR K=1 TO 40:NEXT K
60 NEXT D
70 NEXT P
```

Tabelle 10-1: Charakteristische Eigenschaften der Töne bei den verschiedenen Verzerrungen

Für die Verzerrung angegebene Zahl*	Befehle ohne Auswirkung	Töne mit Obertönen†	Anmerkungen
14	Kommen nicht vor	Kommen nicht vor	Reiner Ton
12	Kommen oft vor	Kommen oft vor	Hohe Töne nur wenig verzerrt
10	Kommen nicht vor	Kommen nicht vor	Reiner Ton
8	Kommen nicht vor	Kommen nicht vor	Tiefe Töne: unregelmäßig Hohe Töne: weißes Rauschen
6	Kommen selten vor	Kommen selten vor	Keine Unterschiede, wenn der für die Tonhöhe angegebene Wert unter 200 liegt.
4	Kommen selten vor	Kommen selten vor	Tiefe Töne: unregelmäßig Hohe Töne: pulsierend
2	Kommen selten vor	Kommen selten vor	Dieselben Töne wie bei 6
0	Kommen selten vor	Kommen nicht vor	Kombination von 4 und 8

* Bei Angabe einer ungeraden Zahl für die Verzerrung wird beim Ein- und Ausschalten einer Stimme ein kurzer Impuls erzeugt.

** Einige Kombinationen von Verzerrung und Tonhöhe löschen sich gegenseitig aus. Bei Ausführung eines solchen Befehls wird kein Ton erzeugt.

† Können bei bestimmten Kombinationen von Verzerrung und Tonhöhe auftreten.

Die Lautstärke

Die vierte auf einen SOUND-Befehl folgende Zahl bestimmt die Lautstärke der angesprochenen Stimme. Die Lautstärke kann also vom Programm aus festgelegt werden. Wenn Sie mit mehreren Stimmen arbeiten, kann die Lautstärke von jeder Stimme unterschiedlich groß sein. Mit dem Lautstärkeregler des Fernsehgerätes bestimmen Sie die Gesamtlautstärke. Wenn dieser Regler überhaupt nicht aufgedreht ist, können Sie natürlich nichts hören.
Die Zahl, die die Lautstärke bestimmt, muß zwischen 0 (Stimme ist nicht zu hören) und 15 (Stimme ist am lautesten) liegen. Die Lautstärke ändert sich linear, das heißt, daß zum Beispiel Lautstärke 8 halb so laut ist wie Lautstärke 15 oder daß Lautstärke 12 ungefähr auf der Mitte zwischen Lautstärke 8 und 15 liegt. Mit Hilfe des folgenden Programmes wird der ganze Lautstärkebereich durchfahren:

```
10 FOR J=-15 TO 15
20 SOUND 0,121,10,ABS(J)
30 NEXT J
40 SOUND 0,0,0,0
```

Wenn Sie ein wenig mit dem SOUND-Befehl gearbeitet haben, werden Sie feststellen, daß die Tonhöhe offensichtlich die Lautstärke beeinflußt. Aus verschiedenen Gründen scheinen die höheren Töne leiser zu sein als die tiefen. Zur Demonstration dieses Effektes dient das folgende Programm:

```
10 FOR J=-50 TO 50
20 SOUND 0,ABS(J),10,8
30 FOR K=1 TO 50:NEXT K
40 NEXT J
50 SOUND 0,0,0,0
```

Befehle zur Tonabschaltung

Wie eine einzelne Stimme abgeschaltet werden kann, haben Sie bereits gesehen. Dazu müssen Sie nur einen SOUND-Befehl geben, bei dem Sie für die Lautstärke eine 0 einsetzen. Es gibt aber in ATARI-BASIC außerdem noch andere Befehle, mit denen Sie alle vier Stimmen gleichzeitig abschalten können. Wenn in einem Programm ein END-Befehl steht, dann werden mit der Ausführung dieses Befehles (also mit dem Programmende) alle vier Stimmen abgeschaltet. Es ist jedoch auch möglich, END direkt einzugeben. Auch bei Ausführung des RUN-Befehls werden alle vier Stimmen abgeschaltet. Wenn Sie also einen Ton eingeschaltet haben und dann ein Programm starten, dann wird dieser Ton abgeschaltet, sofern er nicht vom Programm sofort wieder eingeschaltet wird. Andere Befehle, bei deren Ausführung alle vier Stimmen abgeschaltet werden, sind CLOAD, CSAVE, DOS und NEW. Auch beim Drücken der SYSTEM-RESET-Taste geschieht das, nicht jedoch beim Drücken der BREAK-Taste.

Festlegung der Tondauer

Genauso wichtig wie all die anderen bisher besprochenen Eigenschaften eines Tones ist auch die Tondauer. Mit Hilfe des SOUND-Befehls kann die Tondauer jedoch nicht festgelegt werden, weil es dafür keinen Parameter gibt. Ein einmal eingeschalteter Ton bleibt so lange eingeschaltet, bis er mit Hilfe eines anderen Befehls wieder ausgeschaltet wird. Dies kann natürlich nicht geschehen, solange der Computer andere Befehle ausführt.
Eine Möglichkeit, die Tondauer festzulegen, besteht darin, die Befehle zur Steuerung des Tones mit anderen Befehlen zu verflechten. Auf diese Weise entsteht ein sich regelmäßig verändernder Ton. Das folgende Programm erzeugt einen mit Hilfe des Zufallsgenerators festgelegten Ton, während auf dem Bildschirm eine Zahl ausgegeben wird:

```
10 DIM N$(40)
20 PRINT CHR$(125):REM BILDSCHIRM LOES
CHEN
25 REM PROGRAMMAUSFUEHRUNG HIER FORTSE
TZEN, FALLS EIN FEHLER AUFGETRETEN IST
30 TRAP 20
40 PRINT "GEBEN SIE EINE ZAHL EIN";
50 INPUT N
70 N$=STR$(N)
80 FOR J=1 TO LEN(N$)
90 SOUND 0,64*RND(0)+16,10,10
100 PRINT N$(J,J);
110 SOUND 1,32*RND(0)+8,10,10
120 NEXT J
130 SOUND 0,0,0,0:SOUND 1,0,0,0
140 GOTO 20
```

Nehmen Sie einmal an, daß ein Ton ein- und nach einer gewissen Zeit wieder ausgeschaltet werden soll. Im Programm muß also eine Pause eingefügt werden. Zum Ein- und Ausschalten des Tones verwenden Sie den SOUND-Befehl, aber wie realisieren Sie die Pause? Am einfachsten ist es, wenn Sie hierfür eine FOR-NEXT-Schleife benutzen. Diese Technik wurde bereits in mehreren Programmen angewendet. Das folgende Programm ist ein weiteres Beispiel dafür:

```
10 SOUND 0,47,10,10
20 FOR K=1 TO 100:NEXT K
30 SOUND 0,64,10,10
40 FOR K=1 TO 100:NEXT K
50 GOTO 10
```

Versuche haben ergeben, daß eine leere FOR-NEXT-Schleife in ATARI-BASIC ungefähr 445 mal in der Sekunde durchlaufen wird. Das heißt also, daß eine Schleife mit 100 Durchläufen, wie sie auch im obigen Programm verwendet wurde (Zeile 20 und 40), eine Pause von etwas weniger als einer Viertelsekunde erzeugt. Diese Zeitangabe stimmt jedoch nicht immer. Von Computer zu Computer können hier Unterschiede vorhanden sein. Sie können aber sehr leicht feststellen, wie schnell Ihr eigener ATARI-Computer eine leere FOR-NEXT-Schleife abarbeitet. Alles, was Sie dazu außer dem Computer selbst brauchen, ist eine Uhr. Zuerst geben Sie das folgende Programm ein:

```
10 FOR J=1 TO 35000:NEXT J
```

Als nächstes geben Sie den RUN-Befehl und stoppen die Zeit ab dem Zeitpunkt, an dem Sie die RETURN-Taste gedrückt haben. Nach 30 Sekunden drücken Sie die BREAK-Taste. Mit Hilfe des folgenden Befehls können Sie nun berechnen, wie oft Ihr Computer eine leere FOR-NEXT-Schleife in jeder Sekunde durchläuft:

```
?INT(J/30)
```

Beispiele zur Tonerzeugung

Mit dem ATARI-Computer können sehr realistische Ton- oder auch Geräuscheffekte erzeugt werden. Sie müssen nur für die verschiedenen Stimmen die richtigen Werte für die Tonhöhe, die Verzerrung und die Lautstärke herausfinden und so miteinander kombinieren, daß der gewünschte Effekt entsteht. Das kann manchmal recht schwierig sein, weil es hierfür keine festen Regeln gibt. Hier hilft nur experimentieren. Je mehr Erfahrung Sie haben, desto weniger müssen Sie experimentieren. Wenn Sie erst einmal wissen, wie bestimmte Effekte erzeugt werden, wird es auch einfacher sein, neue Effekte zu realisieren. Wenn Sie einen bestimmten Effekt zu realisieren versuchen, kann es auch einmal vorkommen, daß Sie dabei auf einen ganz anderen Effekt stoßen. Davon sollten Sie sich unbedingt Notizen machen, damit Sie sich, falls es einmal notwendig sein sollte, nicht dieselbe Arbeit noch einmal machen müssen. Zur Einführung dienen die in Bild 10-3 aufgeführten Programme. Probieren Sie sie einmal aus, experimentieren Sie ein wenig damit und versuchen Sie, sie zu verbessern.

```
10 ? "TONDAUER";:INPUT D
99 REM ====JETZT WIRD NACHGEDACHT====
100 FOR J=1 TO D*10
110 SOUND 0,RND(0)*80+50,10,3
120 NEXT J
200 END
```

Bild 10-3: Verschiedene Programme zur Erzeugung von Ton- oder Geräuscheffekten

```
99 REM =======HUEPFENDER BALL=======
100 FOR J=25 TO 1 STEP -1
110 FOR K=1 TO 5
120 SOUND 0,125,14,6
130 NEXT K
140 SOUND 0,0,0,0
150 FOR K=1 TO J*5:NEXT K
160 NEXT J
200 END


99 REM =====LEEREN EINES GLASES=====
100 V=64
110 FOR J=1 TO 30
120 SOUND 0,V-J,10,10
130 SOUND 1,F+J,10,10
140 FOR K=1 TO 30-J:NEXT K
150 SOUND 0,0,0,0
160 SOUND 1,0,0,0
170 FOR K=1 TO 10:NEXT K
180 NEXT J
200 END


99 REM =====FUELLEN EINES GLASES=====
100 FOR J=140 TO 90 STEP -1
110 SOUND 0,J,10,10
120 FOR K=1 TO 20:NEXT K
130 SOUND 0,0,0,0
140 FOR K=1 TO 10:NEXT K
150 NEXT J
200 END


99 REM ===HERUNTERFALLENDES OBJEKT===
100 FOR J=30 TO 200 STEP 3
110 SOUND 0,J,10,J/25
120 FOR K=1 TO J/10:NEXT K
130 NEXT J
140 SOUND 0,20,0,14
150 SOUND 1,255,10,15
160 FOR K=1 TO 100:NEXT K
200 END


99 REM ==========EXPLOSION==========
100 FOR J=-10 TO 10
110 SOUND 0,200,4,10-ABS(J)
120 SOUND 1,255,4,10-ABS(J)
130 SOUND 2,225,4,10-ABS(J)
140 SOUND 3,150,4,10-ABS(J)
160 FOR K=1 TO 200:NEXT K
170 NEXT J
200 END
```

Bild 10-3: Verschiedene Programme zur Erzeugung von Ton- oder Geräuscheffekten (Fortsetzung)

```
10 PRINT "WIEVIEL SCHUSS";
20 INPUT D
99 REM ========GEWEHRSCHUESSE========
100 FOR J=1 TO D
110 SOUND 0,5,0,15
120 FOR K=1 TO 28:NEXT K
130 SOUND 0,0,0,0
140 FOR K=1 TO RND(0)*200:NEXT K
150 NEXT J
200 END


 10 ? "TONDAUER";:INPUT D
 99 REM ========PRSSLUFTHAMMER========
100 FOR J=1 TO D
110 SOUND 0,130+RND(0)*2,2,15
120 FOR K=1 TO 440+RND(0)*100:NEXT K
130 FOR K=1 TO 440+RND(0)*100:NEXT K
140 NEXT J
200 END


10 ? "TONDAUER";:INPUT D
99 REM ============SIRENE============
100 FOR J=1 TO D
110 FOR K=-160 TO 160 STEP 2
120 SOUND 0,ABS(K)+95,10,8
130 FOR L=1 TO 10:NEXT L
140 NEXT K
150 NEXT J
200 END


10 ? "TONDAUER";:INPUT D
99 REM =========MARTINSHORN=========
100 FOR J=1 TO D
110 SOUND 0,47,10,8
120 FOR L=1 TO 100:NEXT L
130 SOUND 0,64,10,8
140 FOR L=1 TO 100:NEXT L
150 NEXT J
200 END


10 ? "TONDAUER";:INPUT D
99 REM =============HUPE=============
100 SOUND 0,121,10,8
110 SOUND 1,128,10,8
120 SOUND 2,8,2,2
130 FOR J=1 TO D*70:NEXT J
200 END
```

Bild 10-3: Verschiedene Programme zur Erzeugung von Ton- oder Geräuscheffekten (Fortsetzung)

```
10 ? "TONDAUER";:INPUT D
99 REM ============SUMMER============
100 SOUND 0,42,2,15
110 FOR J=1 TO D*200:NEXT J
200 END

10 ? "TONDAUER";:INPUT D
99 REM ===========TELEFON===========
100 FOR J=1 TO D
110 SOUND 0,86,10,5
120 SOUND 1,88,10,5
130 SOUND 2,40,2,4
140 FOR K=1 TO 500:NEXT K
150 FOR K=0 TO 3:SOUND K,0,0,0:NEXT K
160 IF J=D THEN GOTO 180
170 FOR K=1 TO 750:NEXT K
180 NEXT J
200 END

10 ? "TONDAUER";:INPUT D
99 REM ============VOEGEL============
100 FOR J=1 TO D*5
110 FOR K=3 TO 10
120 SOUND 0,K,10,8
NEXT K:NEXT J
200 END

10 ? "TONDAUER";:INPUT D
99 REM ===========DAS MEER===========
100 FOR L=1 TO D
110 FOR J=0 TO 45
120 SOUND 0,J,8,4
130 FOR K=1 TO 20+RND(0)*10:NEXT K
140 NEXT J
150 FOR J=45 TO 0 STEP -1
160 SOUND 0,J,8,4
170 FOR K=1 TO 50+RND(0)*30:NEXT K
180 NEXT J:FOR K=1 TO 300+RND(0)*300:N
EXT K
190 NEXT L
200 END

10 ? "TONDAUER";:INPUT D
99 REM =========RAKETENSTART=========
100 FOR L=1 TO D
110 FOR J=0 TO 45
120 SOUND 0,J,8,J/3
140 NEXT J
150 FOR J=45 TO 0 STEP -1
160 SOUND 0,J,8,J/6+6
170 FOR K=1 TO 70+J*3:NEXT K
180 NEXT J
190 NEXT L
200 END
```

Bild 10-3: Verschiedene Programme zur Erzeugung von Ton- oder Geräuscheffekten (Fortsetzung)

11

ZUSAMMENFASSUNG ALLER BASIC-BEFEHLE UND -FUNKTIONEN

In diesem Kapitel wird das Format aller Befehle und Funktionen des ATARI-BASIC beschrieben. Zunächst folgt die Beschreibung der Befehle und dann die der Funktionen; beide in alphabetischer Reihenfolge. Bei den Befehlen finden Sie auch eine Beschreibung der BREAK- und der SYSTEM RESET-Taste. Dabei handelt es sich zwar nicht um Befehle im üblichen Sinne, aber für die Programmausführung sind diese beiden Tasten genauso wichtig wie alle anderen Befehle.
Dieses Kapitel ist ein Leitfaden zur Benutzung aller Befehle und Funktionen. Die in diesem Kapitel aufgeführten Beispiele sollen Ihnen dabei helfen, die BASIC-Befehle richtig zu verwenden. Diese Beispiele können natürlich nicht alle Anwendungsmöglichkeiten eines Befehls abdecken. In den vorhergehenden Kapiteln können Sie jedoch weitere Beispiele finden; viele davon in Form von Programmen.

Verwendung der Befehle und Funktionen in Programmen oder direkt

Alle Befehle und Funktionen können sowohl in Programmen als auch direkt verwendet werden. In manchen Fällen wird jedoch nur eine dieser beiden Möglichkeiten sinnvoll sein.

Die verschiedenen BASIC-Versionen

In diesem Kapitel werden das Format und die Wirkung der Befehle und Funktionen des normalen ATARI-BASIC (auch als Sheperdson BASIC bekannt) beschrieben. Andere BASIC-Versionen wie zum Beispiel das Microsoft BASIC oder BASIC A+ werden damit nicht unbedingt abgedeckt.

Vereinbarungen bezüglich des Formates der Funktionen und Befehle

Zur Darstellung des Formates der BASIC-Funktionen und -Befehle wird nach einem bestimmten Schema vorgegangen. Im Folgenden finden Sie die dabei gültigen Vereinbarungen, zum Beispiel für die Zeichensetzung, die Schreibweise usw.

{}	Hier besteht eine Wahlmöglichkeit. Eine der in den geschweiften Klammern aufgeführten Angaben muß verwendet werden. Die geschweiften Klammern selbst entfallen.
[ ]	Die in eckigen Kalmmern aufgeführten Angaben können zusätzlich gemacht werden. Die eckigen Klammern selbst entfallen.
. . .	Die Fortsetzungspunkte deuten an, daß die davor aufgeführte Angabe wiederholt werden kann. Die Punkte selbst entfallen.
Zeilennummern	Wenn ein Befehl oder eine Funktion in einem Programm verwendet werden soll, muß natürlich eine Zeilennummer angegeben werden.
Andere Zeichen	Alle anderen Zeichen wie zum Beispiel Kommas, Semikolons, Anführungszeichen oder runde Klammern werden direkt übernommen.
GROSS-SCHREIBUNG	Großgeschriebene Worte oder Großbuchstaben werden direkt übernommen.
Kursivschrift	Alles, was in Kursivschrift geschrieben ist, wird nicht direkt übernommen. Es handelt sich hierbei vielmehr um eine Definition, mit deren Hilfe festgelegt wird, wie die Angabe auszusehen hat, die hier eingesetzt werden kann oder muß. Diese Definitionen sind weiter unten oder in den Beschreibungen der einzelnen Befehle und Funktionen aufgeführt. Dort können Sie nachlesen, was Sie an Stelle der jeweiligen Definition einsetzen können.

Allgemeine Definition

Die folgenden kursiv geschriebenen Abkürzungen werden in den Beschreibungen der Befehle und Funktionen verwendet. Mit ihrer Hilfe wird festgelegt, wie eine an Stelle dieser Abkürzung einzusetzende Angabe auszusehen hat. Die Abkürzungen, die Sie hier nicht finden, werden nur für eine bestimmte Funktion oder einen bestimmten Befehl verwendet. Sie werden an der entsprechenden Stelle beschrieben.

Kan	Kanalnummer eines Ein- oder Ausgabekanals. Ein numerischer Ausdruck (*NumAusdr*), in dem allerdings keine Funktionen verwendet werden dürfen und dessen Ergebnis eine ganze Zahl zwischen 1 und 7 sein muß.
Spal	Nummer einer Spalte des Bildschirmes. Ein numerischer Ausdruck, dessen Ergebnis in den Grafikbetriebsarten 0 und 3 zwischen 0 und 39, in den Grafikbetriebsarten 1 und 2 zwischen 0 und 19, in den Grafikbetriebsarten 4 und 5 zwischen 0 und 79, in den Grafikbetriebsarten 6 und 7 zwischen 0 und 159 und in der Grafikbetriebsart 8 zwischen 0 und 319 liegen muß. Nichtganzzahlige Ergebnisse werden gerundet.
Const	Eine numerische oder alphanumerische Konstante. Anführungszeichen dienen hierbei nicht als Begrenzung einer alphanumerischen

	Konstante, sondern sind ein Teil davon.
Ger	Eine alphanumerische Konstante oder Variable. mit deren Hilfe ein Ein- oder Ausgabegerät angesprochen wird. Verwendung finden hier: "C:", "E:", "K:", "P:", "R[*n*]:", "S:" und "D[*n*]:*Filename*[.Ext]".
D[*n*]	Nummer einer Diskettenstation (D, D1, D2, D3 oder D4).
Ausdr	Eine numerische oder boolsche Konstante, Variable oder Funktion, ein numerischer Ausdruck, ein Vergleich oder eine Kombination dieser Möglichkeiten.
Zus	Zusatz zu einem Diskettenfilenamen, bestehend aus einem, zwei oder drei Zeichen. Als Zeichen können die Buchstaben von A bis Z und die Zahlen von 0 bis 9 verwendet werden.
Filename	Ein Diskettenfilename mit einer Länge von einem bis acht Zeichen. Als Zeichen können die Buchstaben von A bis Z und die Zahlen von 0 bis 9 verwendet werden.
EinGer	Eine alphanumerische Konstante oder Variable, mit deren Hilfe ein Eingabegerät angesprochen wird. Verwendung finden hier: "C:", "E:", "K:", "R[*n*]:", "S:" und "D[*n*]:*Filename*[.*Ext*]".
ProZeil	Ein numerischer Ausdruck, mit dessen Hilfe eine Programmzeile eines BASIC-Programmes angesprochen wird. Nichtganzzahlige Zahlen werden gerundet.
SpAdr	Eine numerische Variable, Konstante oder ein Ausdruck, mit dessen Hilfe eine Speicherzelle angesprochen wird. Die Speicherzellen sind von 0 bis 65535 numeriert.
NumAusdr	Eine numerische Konstante, Variable, Funktion oder eine Kombination dieser Möglichkeiten.
NumVar	Eine numerische Variable (keine Feldvariable).
AusGer	Eine alphanumerische Konstante oder Variable, mit deren Hilfe ein Ausgabegerät angesprochen wird. Verwendung finden hier: "C:", "E:", "P:", "R[*n*]:", "S:" und "D[*n*]:*Filename*[.Ext]".
Zeil	Nummer einer Zeile des Bildschirmes. Ein numerischer Ausdruck, dessen Ergebnis in den Grafikbetriebsarten 0, 1 und 3 zwischen 0 und 23, in der Grafikbetriebsart 2 zwischen 0 und 11, in den Grafikbetriebsarten 4 und 5 zwischen 0 und 47, in den Grafikbetriebsarten 6 und 7 zwischen 0 und 95 und in der Grafikbetriebsart 8 zwischen 0 und 191 liegen muß.
AlphaVar	Eine alphanumerische Variable (kein Variablenteil).
Alpha	Eine alphanumerische Konstante oder Variable, ein Variablenteil oder eine Funktion mit einem alphanumerischen Ergebnis.
Var	Eine numerische oder alphanumerische Variable (keine Feldvariable und keine Variablenteil).

Abkürzung der Befehlsworte

In ATARI-BASIC können Sie viele Befehlsworte abkürzen und dadurch beim Eintippen viel Zeit sparen. Wenn Sie zum Beispiel SE. eingeben, dann wird der Computer feststellen, daß es sich hierbei um die Abkürzung für den SETCOLOR-Befehl handelt.
Am Anfang der Beschreibung eines jeden Befehls finden Sie die für diesen Befehl zulässige Abkürzung. Diese Abkürzung können Sie jederzeit an Stelle des ausgeschriebenen Befehlswortes verwenden.

Die BASIC-Befehle

In diesem Abschnitt werden alle BASIC-Befehle beschrieben. Diese Beschreibung umfaßt

die Darstellung des Formates des jeweiligen Befehls in allgemeiner Form und ein oder auch mehrere Anwendungsbeispiele.

BREAK (Break)

Unterbricht die Programmausführung und schaltet um auf den Eingabemodus.

Format: BREAK
Beispiel: BREAK

Beim Drücken der BREAK-Taste wird die Ausführung der BASIC-Anweisung, die der Computer gerade bearbeitet, sofort abgebrochen. Ein Ein- oder Ausgabevorgang kann jedoch nicht zu jedem beliebigen Zeitpunkt unterbrochen werden, was sich manchmal in einer kurzen Verzögerung nach dem Drücken der BREAK-Taste äußert. Allerdings ist es nicht möglich, die Ausführung eines LPRINT-Befehls abzubrechen. Hier müßten Sie an Stelle der BREAK-Taste die SYSTEM RESET-Taste verwenden.
Nach dem Drücken der BREAK-Taste schaltet der Computer auf den Eingabemodus um, aktiviert die Grafikbetriebsart 0 und gibt die Meldung STOPPED AT LINE *Zeilnr* auf dem Bildschirm aus. Für *Zeilnr* wird dabei die Nummer der Zeile eingesetzt, in der das Programm unterbrochen wurde. Wenn die Ausführung des Programmes fortgesetzt werden soll, muß danach der CONT-Befehl gegeben werden. Das Programm wird dann mit der auf *Zeilnr* folgenden Zeile fortgesetzt. Wenn Sie jedoch vor Eingabe des CONT-Befehls einen anderen Befehl geben, kann das Programm nicht fortgesetzt werden.
Falls Sie die BREAK-Taste im Eingabemodus drücken, wird die logische Zeile, mit der Sie gerade arbeiten, verlassen. Der Computer springt danach an den Anfang der nächsten logischen Zeile.
Beim Drücken der BREAK-Taste wird weder die Tonerzeugung beeinflußt noch werden irgendwelche geöffneten Ein- oder Ausgabekanäle geschlossen.

BYE (B.)

Umschalten von BASIC in die Memo-Pad-Betriebsart.

Format: BYE
Beispiele: BYE
B.

Der zum Speichern von Variablen oder eines BASIC-Programmes verwendete Speicherbereich wird bei der Ausführung dieses Befehles nicht verändert. Wenn Sie durch Drücken der SYSTEM RESET-Taste wieder auf das BASIC umschalten, werden alle Programmzeilen, die vor Ausführung des BYE-Befehls im Speicher standen, noch immer vorhanden sein. Auch die Variablentabelle ist nicht verändert worden. Auf ein Disk Operating System und/oder ein Programm zur Bedienung einer RS-232-Schnittstelle, das gebootet war, kann, nachdem wieder auf das BASIC umgeschaltet worden ist, noch immer zugegriffen werden.

CLOAD (CLOA.)

Lädt ein auf Kassette gespeichertes Programm mit Hilfe des Programmrecorders und legt es im Speicher ab.

Format: CLOAD

Beispiele: CLOAD
CLOA.

Nach der Eingabe dieses Befehls wird zunächst Kanal 7 für Eingabe von dem Programmrecorder geöffnet. Falls Kanal 7 bereits belegt ist, wird eine Fehlermeldung ausgegeben. Gleichzeitig wird der Kanal geschlossen. Dadurch wird vermieden, daß beim nächsten Versuch, ein Programm mit CLOAD zu laden, erneut ein Fehler auftritt.
Bei der Ausführung des CLOAD-Befehls ertönt zunächst ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie nach einer kleinen Pause eine oder mehrere schnelle Tonfolgen hören: Das Programm wird geladen.
Das Format eines Programmes, das mit CLOAD geladen werden soll, ist nicht beliebig. Es muß mit Hilfe des CSAVE- oder des SAVE-Befehls gespeichert worden sein. Ein Programm, das unter Verwendung des LIST-Befehls im ATASCII-Kode gespeichert wurde, kann mit CLOAD nicht geladen werden.
Während des Ladevorganges wird außerdem die gerade im Speicher befindliche Variablentabelle gelöscht und durch die des Programmes, das gerade geladen wird, ersetzt.

CLOAD und NEW

Im CLOAD-Befehl ist der NEW-Befehl gewissermaßen enthalten. Noch bevor der Kontrollton ertönt, werden alle Programmzeilen und Variablen aus dem Speicher gelöscht. Wenn Sie nach der Ausgabe des Kontrolltones die BREAK-Taste drücken, wird die Ausführung des CLOAD-Befehls zwar unterbrochen, aber trotzdem wird ein eventuell im Speicher vorhandenes Programm inzwischen gelöscht worden sein. Auch von dem neuen Programm wurde zu diesem Zeitpunkt noch nichts geladen.
Nachdem der Ladevorgang beendet worden ist (ganz gleich, ob erfolgreich oder nicht erfolgreich beziehungsweise vollständig oder nicht vollständig), wird jegliche Tonerzeugung abgeschaltet und alle Ein-/Ausgabekanäle bis auf Kanal 0 geschlossen. Hierbei ist zu beachten, daß auch Kanal 6, der in einigen Grafikbefehlen enthalten ist, geschlossen wird.

Abbruch von CLOAD

Die Ausführung des CLOAD-Befehls kann jederzeit durch Drücken der SYSTEM RESET-Taste abgebrochen werden. Die BREAK-Taste erfüllt denselben Zweck, nur wird sie in den ersten 20 Sekunden nach Beginn des Ladevorganges, während denen der Programmrecorder den vor jedem Programm befindlichen Leader liest, nicht funktionieren.

CLOSE (CL.)

Schließt einen Ein-/Ausgabekanal.

Format: CLOSE #*Kan*
Beispiele: CLOSE #1
CL. #GERAETA

Bevor ein für ein bestimmtes Gerät geöffneter Ein-/Ausgabekanal mit Hilfe eines OPEN-Befehls einem anderen Gerät zugewiesen werden kann, muß dieser Kanal geschlossen

werden. Der CLOSE-Befehl bleibt wirkungslos, wenn Sie einen Kanal schließen, der gar nicht geöffnet war. Eine Fehlermeldung wird in diesem Fall nicht ausgegeben.
Wenn ein Kanal für Ausgabe auf dem Programmrecorder oder der Diskettenstation geöffnet ist, kann es vorkommen, daß Daten, die eigentlich schon gespeichert sein sollten, noch im Hauptspeicher des Computers stehen und darauf warten, ausgegeben zu werden. Das liegt daran, daß die Ausgabe bei diesen Geräten in 128-Byte-Blöcken erfolgt und der Computer die auszugebenden Daten solange sammelt, bis diese Grenze erreicht ist. Wenn der Kanal geschlossen wird, werden die Daten, gefolgt von einem End-of-File-Record, jedoch auch dann ausgegeben, wenn die 128 Bytes noch nicht erreicht sind.
Bei der Ausführung des END-Befehls werden alle offenen Kanäle außer Kanal 0 geschlossen. Kanal 0 wird vom BASIC zur Bedienung der Tastatur und des Bildschirmes benutzt.

CLR

Weist allen numerischen Variablen eine 0 zu. Eine eventuell vorhandene Dimensionierung für Feld- oder alphanumerische Variablen gilt nicht mehr. Der Zeiger zum Lesen von Daten wird an den Anfang des ersten DATA-Befehls gesetzt.

Format: CLR
Beispiel: CLR

Bei Ausführung des CLR-Befehls werden die Variablennamen nicht, wie das beim NEW-Befehl der Fall ist, aus der Variablentabelle gelöscht. Es ist also nicht möglich, mit Hilfe des CLR-Befehls für neue Variablennamen Platz zu schaffen, wenn in einem Programm ein Fehler auftritt, weil mehr als 128 Variablen verwendet wurden.
Allerdings macht CLR eine eventuell vorhandene Dimensionierung für Feld- oder alphanumerische Variablen ungültig. So ist es möglich, nach der Ausführung eines CLR-Befehls diese Variablen neu zu dimensionieren und zwar entsprechend dem zur Verfügung stehenden freien Speicherplatz.

COLOR (C.)

Bestimmt, welches Farbregister bei der Ausführung des nächsten PLOT- oder DRAWTO-Befehls zur Festlegung der Zeichenfarbe verwendet wird. In den Grafikbetriebsarten 0, 1 und 2 bestimmt COLOR, welches Zeichen bei der Ausführung des nächsten PLOT- oder DRAWTO-Befehls dargestellt wird.

Format: COLOR *NumAusdr*
Beispiele: COLOR 1
C. ASC("Z")
COLOR C1+A3/3

Mit Hilfe von *NumAusdr* wird ausgewählt, welches Farbregister bei Ausführung des nächsten PLOT- oder DRAWTO-Befehls zur Festlegung der Zeichenfarbe verwendet wird. Der Wert von *NumAusdr* muß zwischen 0 und 65535 liegen. Nichtganzzahlige Zahlen werden gerundet.
Es gibt fünf Farbregister, die mit den Zahlen von 0 bis 4 gekennzeichnet sind. Tabelle 11 – 1 können sie entnehmen, welchen Wert *NumAusdr* in einer bestimmten Grafikbetriebsart haben muß, damit ein bestimmtes Farbregister angesprochen wird. Dieser Tabelle können Sie zum Beispiel entnehmen, daß mit Hilfe des Befehls COLOR 2 in den Grafikbetriebsarten 3, 5 und 7 das Farbregister 1 ausgewählt wird.

Festlegung der Farben

In jedem Farbregister ist eine Information darüber gespeichert, welche Farbe in welcher Helligkeitsstufe dargestellt werden soll. Mit Hilfe des SETCOLOR-Befehls können diese Informationen verändert werden. Die Farbregister haben gewisse Voreinstellungen (siehe Tabelle 11-2). Diese Voreinstellungen gelten nach dem Einschalten des Computers, nach dem Drücken der SYSTEM RESET-Taste oder nach der Ausführung eines DOS- oder GRAPHICS-Befehls.

Tabelle 11-1: Auswahl der Farbregister

Nummer des Farbregisters beim SETCOLOR-Befehl	**Wert von *NumAusdr* beim COLOR *NumAusdr*-Befehl in den verschiedenen Grafikbetriebsarten***		
	3, 5, 7	**4, 6**	**8****
0	1	1	–
1	2	–	1
2	3	–	0
3	–	–	–
4	0	0	–

* In den Grafikbetriebsarten 0, 1 und 2 bestimmt *NumAusdr*, welches Zeichen bei Ausführung des nächsten PLOT- oder DRAWTO-Befehls gezeichnet wird (siehe Tabellen 11-3 und 11-4).

** In der Grafikbetriebsart 8 kann mit Hilfe des COLOR-Befehls nicht die Farbe, sondern nur die Helligkeit festgelegt werden. Die Farbe bestimmt hier grundsätzlich Farbregister 2.

Tabelle 11-2: Voreinstellungen der Farbregister

Nummer des Farbregisters	**Farbe**	**Helligkeit**	**Auf dem Bildschirm dargestellte Farbe**
0	2	8	Orange
1	12	10	Grün
2	9	4	Dunkelblau
3	4	6	Rot
4	0	0	Schwarz

Verwendung des COLOR-Befehls in der Grafikbetriebsart 0

In der Grafikbetriebsart 0 werden keine Punkte, sondern Zeichen dargestellt. Hier wird mit Hilfe des COLOR-Befehls festgelegt, welche Zeichen bei Verwendung des PLOT- oder DRAWTO-Befehls auf dem Bildschirm ausgegeben werden. In Tabelle 11-3 finden Sie eine Zusammenstellung der Zahlen, die für *NumAusdr* eingesetzt werden können und der Zeichen, die damit dargestellt werden. Der Wert von *NumAusdr* darf zwischen 0 und 65535 liegen. Wenn er jedoch 255 übersteigt, wird solange 256 subtrahiert, bis wieder ein Wert herauskommt, der zwischen 0 und 255 liegt.

Verwendung des COLOR-Befehls in den Grafikbetriebsarten 1 und 2

Die Grafikbetriebsarten 1 und 2 sind der Grafikbetriebsart 0 sehr ähnlich. Hier gibt es jedoch zwei Zeichensätze mit jeweils 64 Zeichen. Normalerweise wird hauptsächlich der erste Zeichensatz verwendet. In ihm sind Großbuchstaben, Zahlen und Satzzeichen enthalten. Der zweite Zeichensatz enthält Kleinbuchstaben und Grafikzeichen. Nach dem Ein-

Tabelle 11-3: Die Verwendung des COLOR *NumAusdr*-Befehls in der Grafikbetriebsart 0
(Einsetzen einer bestimmten Zahl für *NumAusdr* zur Darstellung *eines* bestimmten Zeichens)

Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen	Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen	Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen	Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen	Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen	Einzusetzender Wert zur norm./invert. Darstellung	Dargestelltes Zeichen
0/128		23/151		46/174		69/197		92/220		115/243	
1/129		24/152		47/175		70/198		93/221		116/244	
2/130		25/153		48/176		71/199		94/222		117/245	
3/131		26/154		49/177		72/200		95/223		118/246	
4/132		27/---		50/178		73/201		96/224		119/247	
5/133		28/156		51/179		74/202		97/225		120/248	
6/134		29/157		52/180		75/203		98/226		121/249	
7/135		30/158		53/181		76/204		99/227		122/250	
8/136		31/159		54/182		77/205		100/228		123/251	
9/137		32/160		55/183		78/206		101/229		124/252	
10/138		33/161		56/184		79/207		102/230		125/---	clr scrn
11/139		34/162		57/185		80/208		103/231		126/254	
12/140		35/163		58/186		81/209		104/232		127/255	
13/141		36/164		59/187		82/210		105/233		---/155	EOL
14/142		37/165		60/188		83/211		106/234		---/253	
15/143		38/166		61/189		84/212		107/235			
16/144		39/167		62/190		85/213		108/236			
17/145		40/168		63/191		86/214		109/237			
18/146		41/169		64/192		87/215		110/238			
19/147		42/170		65/193		88/216		111/239			
20/148		43/171		66/194		89/217		112/240			
21/149		44/172		67/195		90/218		113/241			
22/150		45/173		68/196		91/219		114/242			

schalten des Computers, nach Drücken der SYSTEM RESET-Taste oder nach der Ausführung eines GRAPHICS-Befehls ist der erste Zeichensatz aktiviert. Bei Ausführung des Befehls POKE 756,226 wird auf den zweiten Zeichensatz umgeschaltet. Mit POKE 756,224 wird wieder der erste Zeichensatz aktiviert.

In Tabelle 11-4 finden Sie eine Zusammenstellung der Zahlen, die für *NumAusdr* eingesetzt werden können und der Zeichen des ersten und zweiten Zeichensatzes, die damit in den Grafikbetriebsarten 1 und 2 dargestellt werden. Wie Sie sehen, können zur Erzeugung eines bestimmten Zeichens vier verschiedene Zahlen eingesetzt werden. Es handelt sich zwar jedesmal um ein und dasselbe Zeichen, aber dieses Zeichen wird jedesmal in einer

Tabelle 11-4: Die Verwendung des COLOR *NumAusdr*-Befehls in den Grafikbetriebsarten 1 und 2 (Einsetzen einer bestimmten Zahl für *NumAusdr* zur Darstellung eines bestimmten Zeichens in einer bestimmten Farbe)

Einzusetzender Wert zur Auswahl des Farbregisters				Dargestelltes Zeichen*		Einzusetzender Wert zur Auswahl des Farbregisters				Dargestelltes Zeichen*	
0	1	2	3	Std.	Alt.	0	1	2	3	Std.	Alt.
32**	0	160	128			64	96	192	224	@	
33	1	161	129	!		65	97	193	225	A	a
34	2	162	130	"		66	98	194	226	B	b
35	3	163	131	#		67	99	195	227	C	c
36	4	164	132	$		68	100	196	228	D	d
37	5	165	133	%		69	101	197	229	E	e
38	6	166	134	&		70	102	198	230	F	f
39	7	167	135	'		71	103	199	231	G	g
40	8	168	136	(		72	104	200	232	H	h
41	9	169	137	)		73	105	201	233	I	i
42	10	170	138	*		74	106	202	234	J	j
43	11	171	139	+		75	107	203	235	K	k
44	12	172	140	,		76	108	204	236	L	l
45	13	173	141	-		77	109	205	237	M	m
46	14	174	142	.		78	110	206	238	N	n
47	15	175	143	/		79	111	207	239	O	o
48	16	176	144	0		80	112	208	240	P	p
49	17	177	145	1		81	113	209	241	Q	q
50	18	178	146	2		82	114	210	242	R	r
51	19	179	147	3		83	115	211	243	S	s
52	20	180	148	4		84	116	212	244	T	t
53	21	181	149	5		85	117	213	245	U	u
54	22	182	150	6		86	118	214	246	V	v
55	23	183	151	7		87	119	215	247	W	w
56	24	184	152	8		88	120	216	248	X	x
57	25	185	153	9		89	121	217	249	Y	y
58	26	186	154	:		90	122	218	250	Z	z
59	27	187	-***	;		91	123	219	251	[	
60	28	188	156	<		92	124	220	252	\	
61	29	189	157	=		93	-***	221	253	]	
62	30	190	158	>		94	126	222	254	^	
63	31	191	159	?		95	127	223	255	_	

Anmerkung:

* Zur Auswahl des ersten Zeichensatzes geben Sie den Befehl POKE 756.224, zur Auswahl des zweiten POKE 756.226.

** Bei Angabe von 155 wird dasselbe Zeichen in derselben Farbe dargestellt wie bei der Angabe von 32.

*** Die Darstellung dieses Zeichens in dieser Farbe ist nicht möglich.

anderen Farbe dargestellt. Der Wert von *NumAusdr* darf zwischen 0 und 65535 liegen. Wenn er jedoch 255 übersteigt, wird solange 256 subtrahiert, bis wieder ein Wert herauskommt, der zwischen 0 und 255 liegt.

Verwendung des COLOR-Befehls in den Grafikbetriebsarten 3 bis 7

In den Grafikbetriebsarten 3 bis 7 wird mit Hilfe von *NumAusdr* ausgewählt, welches Farbregister bei der Ausführung des nächsten PLOT- oder DRAWTO-Befehls zur Festlegung der Zeichenfarbe und -helligkeit verwendet wird. Tabelle 11-1 können Sie entnehmen, welche Farbregister in den verschiedenen Grafikbetriebsarten benutzt werden können. Der Wert von *NumAusdr* darf zwischen 0 und 65535 liegen. Wenn er jedoch 3 übersteigt, wird solange 4 subtrahiert, bis wieder ein Wert herauskommt, der zwischen 0 und 3 liegt.

Verwendung des COLOR-Befehls in der Grafikbetriebsart 8

In der Grafikbetriebsart 8 wird die Farbe aller Punkte und Linien sowie des Hintergrundes durch das Farbregister 2 festgelegt. Mit Hilfe des COLOR-Befehls wird nicht die Farbe, sondern nur die Helligkeit der Punkte und Linien beeinflußt. Mit *NumAusdr* wird das Farbregister ausgewählt, das die Helligkeit bestimmt (siehe Tabelle 11-1). Der Wert von *NumAusdr* darf zwischen 0 und 65535 liegen. Wenn er jedoch 3 übersteigt, wird solange 4 subtrahiert, bis wieder ein Wert herauskommt, der zwischen 0 und 3 liegt.

COM

Stellt Speicherplatz für Feld- oder alphanumerische Variablen bereit.

Format: COM{*AlphaVar (NumAusdr)* / *NumVar (NumAusdr [,NumAusdr])*} [,{*AlphaVar (NumAusdr)* / *NumVar (NumAusdr [NumAusdr])*}...]

Beispiele: COM A$(24), FELD1(25), FELD2(5,5)
COM NAME$(30), ADDR1$(30), ADDR2$(30), ADDR3$(30)

Der COM-Befehl entspricht in allen Einzelheiten dem DIM-Befehl.

CONT (CON.)

Setzt die Programmausführung nach einer Unterbrechung fort.

Format: CONT
Beispiel: CONT

Ein Programm, das durch Drücken der BREAK-Taste oder beim Auftreten eines STOP- oder END-Befehls unterbrochen wurde, kann mit Hilfe des CONT-Befehls fortgesetzt werden.
Die Programmausführung wird mit der Zeile fortgesetzt, die unmittelbar auf die Zeile folgt, wo die Unterbrechung auftrat. Wenn die Unterbrechung innerhalb einer Zeile auftrat, in der mehrere Befehle stehen, dann heißt das, daß die auf die Unterbrechung folgenden Befehle dieser Zeile bei der Fortsetzung des Programmes nicht ausgeführt werden. Stattdessen wird mit der Ausführung der nächsten Zeile begonnen.

Der CONT-Befehl und die BREAK-Taste

Die Ausführung einiger Befehle, zum Beispiel des INPUT- oder des LIST-Befehls, dauert eine gewisse Zeit. Wenn Sie während der Ausführung eines solchen Befehls die BREAK-Taste drücken, dann wird dieser Befehl unterbrochen. Ein darauf folgender CONT-Befehl setzt die Programmausführung mit dem ersten Befehl der nächsten Programmzeile fort. Der unterbrochene Befehl selbst wird nicht weiter bearbeitet.
Wenn Sie die BREAK-Taste zufällig genau in dem Augenblick betätigen, wenn die Ausführung einer Zeile beendet worden ist und die nächste Zeile noch nicht begonnen wurde, dann wird die Programmausführung sofort gestoppt, also noch bevor mit der Ausführung der nächsten Zeile begonnen wurde. Wenn Sie nun einen CONT-Befehl geben, dann wird diese Programmzeile übersprungen und die Programmausführung mit der darauffolgenden Zeile fortgesetzt.

Verwendung des CONT-Befehls, ohne daß vorher ein Programm Programm unterbrochen wurde

Sie können natürlich auch einen CONT-Befehl geben, ohne daß vorher ein laufendes Programm unterbrochen wurde. Der Computer bearbeitet diesen CONT-Befehl so, als wenn das im Speicher befindliche Programm nach der Ausführung des ersten Befehls der ersten Programmzeile unterbrochen worden wäre. Die Programmausführung beginnt dann also mit der zweiten Programmzeile.

Verwendung des CONT-Befehls nach dem Auftreten eines Fehlers

Auch beim Auftreten eines Fehlers wird die Programmausführung unterbrochen und auch in diesem Fall können Sie das Programm mit Hilfe des CONT-Befehls fortsetzen, aber es wird dann weder der Befehl, der den Fehler verursachte, noch irgendein anderer Befehl derselben Programmzeile bearbeitet. Aus diesem Grund müssen Sie sehr vorsichtig sein, wenn Sie den CONT-Befehl nach dem Auftreten eines Fehlers verwenden. Der oder die Befehle, die nicht bearbeitet werden, könnten sehr wichtig sein. Ihr Fehlen kann im weiteren Verlauf des Programmes zu erheblichen Problemen führen. Auf jeden Fall wäre es eine bessere Lösung, wenn Sie zuerst die Ursache des Fehlers beheben und die Programmausführung dann mit einem GOTO-Befehl fortsetzen.

Verwendung des CONT-Befehls nach Betätigen der SYSTEM RESET-Taste

Auch nach Betätigen der SYSTEM RESET-Taste können Sie ein Programm mit Hilfe des CONT-Befehls fortsetzen. Die Programmausführung wird mit der Programmzeile fortgesetzt, die unmittelbar auf die Zeile folgt, die durch Drücken der SYSTEM RESET-Taste unterbrochen wurde. Es ist allerdings ziemlich unwahrscheinlich, daß ein Programm nach einem SYSTEM RESET noch ordnungsgemäß arbeitet.

Verwendung des CONT-Befehls in einem Programm

Der CONT-Befehl kann sowohl in einem Programm als auch im Eingabemodus verwendet werden. Normalerweise wird man allerdings nur letzteres tun, weil der CONT-Befehl in einem Programm absolut keine Funktion hat und nur zusätzliche Rechenzeit und zusätzlichen Speicherplatz in Anspruch nimmt.

CSAVE (CS.)

Legt ein im Speicher befindliches Programm mit Hilfe des Programmrecorders auf Kassette ab.

Format: CSAVE
Beispiele: CSAVE
CS.

Nach der Eingabe dieses Befehls wird zunächst Kanal 7 für Ausgabe auf den Programmrecorder geöffnet. Falls Kanal 7 bereits belegt ist, wird eine Fehlermeldung ausgegeben. Gleichzeitig wird der Kanal geschlossen. Dadurch wird vermieden, daß beim nächsten Versuch, ein Programm mit CSAVE zu speichern, erneut ein Fehler auftritt.
Bei der Ausführung des CSAVE-Befehls ertönt zunächst zweimal ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die RECORD- und die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Daraufhin wird zunächst jegliche Tonerzeugung abgeschaltet. Danach werden Sie, wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, etwa 20 Sekunden lang einen gleichmäßigen hohen Ton hören, auf den eine oder mehrere schnelle Tonfolgen folgen: Das Programm wird gespeichert.
Im Gegensatz zu anderen Befehlen erfolgt die Speicherung eines Programmes bei Verwendung des CSAVE-Befehls nicht im ATASCII-Kode. Hier wird außer dem Programm auch die gerade im Speicher befindliche Variablentabelle mitgespeichert. Nur mit Hilfe des CLOAD-Befehls kann ein mit CSAVE gespeichertes Programm geladen werden. Es ist hingegen nicht möglich, ein mit CSAVE gespeichertes Programm mit Hilfe des ENTER- oder LOAD-Befehls zu laden.

Abbruch von CSAVE

Die Ausführung des CSAVE-Befehls kann jederzeit durch Drücken der SYSTEM RESET- oder der BREAK-Taste abgebrochen werden. Das Programm ist dann natürlich nicht vollständig gespeichert worden und kann auch nicht mit Hilfe des CLOAD-Befehls geladen werden.

DATA (D.)

Dient zur Festlegung von Werten, die später mit Hilfe des READ-Befehls Variablen zugewiesen werden können.

Format: DATA *Const* [,*Const* ...]
Beispiele: DATA Sonntag, Montag, Dienstag,
Mittwoch, Donnerstag, Freitag, Samstag
D. 100, -89, 1.414E-2
DATA 2+2
D. GRÜNKOHL, WEISSKRAUT, "SPINAT",

Der DATA-Befehl dient zur Festlegung von Werten von numerischen oder alphanumerischen Variablen oder auch von beiden. Die Werte werden den Variablen mit Hilfe des READ-Befehls zugewiesen. Zur Trennung der Konstanten voneinander werden Kommas verwendet. DATA-Befehle können an jeder beliebigen Stelle eines Programmes eingefügt werden. Um auf sie mit Hilfe des READ-Befehls zugreifen zu können, müssen sie nicht un-

bedingt vorher vom Programm durchlaufen worden sein. Auf einen DATA-Befehl darf nicht innerhalb derselben Programmzeile ein anderer Befehl folgen.

Angabe von alphanumerischen Konstanten in einem DATA-Befehl

Da zur Trennung der Konstanten Kommas verwendet werden, können alphanumerische Konstanten keine Kommas enthalten. Alle anderen Zeichen, einschließlich der Anführungszeichen und des Leerzeichens, können Teil einer alphanumerischen Konstanten sein. Eine alphanumerische Konstante kann sogar ausschließlich aus Leerzeichen bestehen. Falls zwei Kommas direkt aufeinander folgen, dann ist die entsprechende Konstante 0 Zeichen lang. Dasselbe gilt, wenn am Ende eines DATA-Befehls ein Komma steht.

Angabe von numerischen Konstanten in einem DATA-Befehl

Die Angabe von numerischen Konstanten kann in der normalen oder in der Exponentialschreibweise erfolgen. Die Länge einer numerischen Konstanten kann allerdings nicht 0 sein. Wenn zwei Kommas direkt aufeinander folgen und an dieser Stelle mit Hilfe eines READ-Befehls einer numerischen Variablen ein Wert zugewiesen werden soll, wird ein Fehler auftreten.
Arithmetische Ausdrücke werden nicht berechnet, sondern stattdessen, wie alphanumerische Konstanten behandelt. Den Ausdruck 2+2 sieht der Computer zum Beispiel als eine drei Zeichen lange alphanumerische Konstante an und nicht als eine numerische Konstante mit dem Wert 4.

Verwendung des DATA-Befehls im Eingabemodus

Bei der Verwendung des DATA-Befehls im Eingabemodus tritt zwar kein Fehler auf, aber auf die in diesem DATA-Befehl aufgeführten Konstanten kann nicht mit Hilfe des READ-Befehls zugegriffen werden.

DEG (DE.)

Schaltet vom Bogenmaß auf Grad um.

Format: DEG
Beispiele: DEG
DE.

Nach der Ausführung des DEG-Befehls betrachtet das BASIC die Argumente aller nun folgenden trigonometrischen Funktionen als Gradangaben. Mit Hilfe des RAD-Befehls kann wieder auf das Bogenmaß umgeschaltet werden. Auch nach dem Einschalten des Computers, nach Drücken der SYSTEM RESET-Taste oder nach der Ausführung eines NEW- oder RUN-Befehls wird im Bogenmaß gerechnet.

DIM

Stellt Speicherplatz für Feld- oder alphanumerische Variablen bereit.

Format: DIM{*AlphaVar (NumAusdr)* / *NumVar (NumAusdr [,NumAusdr])*} [{*AlphaVar (NumAusdr)* / *NumVar (NumAusdr [NumAusdr])*}...]

Beispiele: DIM A$(24), FELD1(25), FELD2(5,5)
DIM NAME$(30), ADDR1$(30), ADDR2$(30), ADDR3$(30)

Bevor Feld- oder alphanumerische Variablen benutzt werden können, müssen sie dimensioniert werden. Felder dürfen in ATARI-BASIC ein- oder zweidimensional sein; alphanumerische Variablen können ein oder mehrere Zeichen lang sein.

Feldvariablen

Bei der Dimensionierung eines Feldes wird für jedes Element dieses Feldes Speicherplatz reserviert. *NumAusdr* bestimmt die Größe des Feldes, oder, mit anderen Worten, die maximale Größe der Zahl, die bei der Angabe eines Feldelementes als Index verwendet werden kann. Wenn die für den Index angegebene Zahl größer als diese obere Grenze oder kleiner als 0 ist, wird ein Fehler auftreten. *NumAusdr* wird gerundet, wenn es sich dabei um eine nichtganzzahlige Zahl handelt.

Alphanumerische Variablen

Mit Hilfe des DIM-Befehls wird die maximale Länge einer alphanumerischen Variablen festgelegt. Sie entspricht dem Wert von *NumAusdr.* Auch hier wird *NumAusdr* gerundet, wenn es sich dabei um eine nichtganzzahlige Zahl handelt. Die tatsächliche Länge der alphanumerischen Variablen kann allerdings, wenn sie später im Programm verwendet wird, zwischen 0 und der festgelegten Maximallänge liegen.

Die maximale Größe von Feld- und alphanumerischen Variablen

Eine alphanumerische Variable darf maximal 32767 Zeichen lang sein. Die maximale Größe der Feld- und alphanumerischen Variablen wird außerdem durch den im Moment der Ausführung des DIM-Befehls zur Verfügung stehenden freien Speicherplatz begrenzt. Nach einer Dimensionierung kann die Größe von Feld- und alphanumerischen Variablen nur nach der Ausführung eines CLR-Befehls geändert werden, wobei die geltende Dimensionierung aufgehoben wird. Wenn eine bereits dimensionierte Variable ein zweites Mal dimensioniert wird, tritt ein Fehler auf, auch dann, wenn die angegebene Größe der Variablen beim zweiten DIM-Befehl genauso groß ist wie beim ersten.

DOS (DO.)

Dient zur Aktivierung des DOS-Menues.

Format: DOS
Beispiel: DOS
DO.

Nach der Ausführung des DOS-Befehls erscheint das Menue des Disk Operating System mit seinen 15 Hilfsfunktionen auf dem Bildschirm. Wenn vorher kein Disk Operating System gebootet wurde, schaltet der Computer auf die Memo-Pad-Betriebsart um (siehe auch Beschreibung des BYE-Befehls).
Bei der Ausführung eines DOS-Befehls wird zunächst der Bildschirm gelöscht, der Wert der Farbregister entsprechend den Voreinstellungen (siehe Tabelle 11-2) geändert, jegliche Tonerzeugung abgeschaltet und alle Ein/Ausgabekanäle bis auf Kanal 0 geschlossen. Hierbei ist zu beachten, daß auch Kanal 6, der bei der Ausführung einiger Grafikbefehle

benutzt wird, geschlossen wird.
Im Augenblick gibt es zwei verschiedene Versionen des Disk Operating System, nämlich DOS 1.0 und DOS 2.0S. Die Menues dieser beiden Versionen unterscheiden sich ein wenig. In Tabelle 11-5 sind beide nebeneinander aufgeführt. In Kapitel 7 finden Sie für jede einzelne Funktion der Menues eine ausführliche Beschreibung.

Tabelle 11-5: Die Hilfsfunktionen der DOS-Menues

Disk Operating System	
Version 1.0	**Version 2.0S**
A. Disk Directory	A. Disk Directory
B. Run Cartridge	B. Run Cartridge
C. Copy File	C. Copy File
D. Delete File(s)	D. Delete File(s)
E. Rename File	E. Rename File
F. Lock File	F. Lock File
G. Unlock File	G. Unlock File
H. Write DOS File	H. Write DOS Files
I. Format Disk	I. Format Disk
J. Duplicate Disk	J. Duplicate Disk
K. Binary Save	K. Binary Save
L. Binary Load	L. Binary Load
M. Run at Address	M. Run at Address
N. Define Device	N. Create MEM.SAV
O. Duplicate File	O. Duplicate File

Verwendung des DOS-Befehls unter DOS 1.0

Wenn Sie mit DOS 1.0 arbeiten und den DOS-Befehl geben, erscheint das DOS-Menue sofort auf dem Bildschirm. Sie können nun eine der DOS-Funktionen wählen oder ins BASIC zurückkehren. In Kapitel 7 finden Sie für jede Funktion des DOS-Menues eine ausführliche Beschreibung.
Um wieder ins BASIC zurückzukehren, drücken Sie die SYSTEM RESET-Taste oder Sie wählen Funktion B des DOS-Menues. Ein eventuell vor Ausführung des DOS-Befehls im Speicher befindliches BASIC-Programm wird noch vorhanden sein, sofern Sie nicht Funktion J (DUPLICATE DISK) oder Funktion O (DUPLICATE FILE) des DOS-Menues benutzt haben. Sobald Sie eine dieser beiden Funktionen benutzen, werden alle im Speicher befindlichen Programme und Daten gelöscht.

Verwendung des DOS-Befehls unter DOS 2.0S

Nachdem Sie den DOS-Befehl gegeben haben, erscheint hier nicht sofort das DOS-Menue auf dem Bildschirm, weil der Computer zunächst von der in Laufwerk 1 eingelegten Diskette das File mit den Namen DUP.SYS laden muß. Das dauert etwa 10 Sekunden. Falls das File nicht gefunden wird, bleibt das BASIC aktiviert.
Bevor DUP.SYS geladen wird, überprüft der Computer, ob sich auf der in Laufwerk 1 eingelegten Diskette ein MEM.SAV-File befindet. Wenn dem so ist, wird der Inhalt des Speicherbereiches, den später DUP.SYS belegt, mit Hilfe dieses Files zwischengespeichert. In diesem Speicherbereich befindet sich der Anfang eines eventuell im Speicher befindlichen BASIC-Programmes und, falls es gebootet wurde, das Programm zur Bedienung der RS-232-Schnittstelle. Um diesen Speicherbereich auf Diskette abzulegen, werden noch einmal etwa 20 Sekunden benötigt.

Nachdem all diese Aufgaben erledigt worden sind, wird das DOS-Menue auf dem Bildschirm angezeigt. Sie können nun eine der DOS-Funktionen wählen oder ins BASIC zurückkehren. In Kapitel 7 finden Sie für jede Funktion des DOS-Menues eine ausführliche Beschreibung.
Um wieder ins BASIC zurückzukehren, drücken Sie die SYSTEM RESET-Taste oder Sie wählen Funktion B des DOS-Menues. Ein eventuell vor Aufruf des DOS-Menues im Speicher befindliches BASIC-Programm oder ein Programm zur Bedienung einer RS-232-Schnittstelle oder auch beides wird noch vorhanden sein, wenn das MEM.SAV-File zur Zwischenspeicherung dieser Programme beziehungsweise Programmteile benutzt wurde. Zum erneuten Laden dieses Speicherbereiches werden etwa 7 Sekunden benötigt. Wenn Sie jedoch nach dem Aufruf von Funktion C (COPY FILE) oder Funktion O (DUPLICATE FILE) des DOS-Menues die Frage, ob der Programmspeicher zum Kopieren benutzt werden darf, bejaht haben oder wenn Sie Funktion J des DOS-Menues (DUPLICATE DISK) gewählt haben, dann können auch mit Hilfe des MEM.SAV-Files die vorher im Speicher befindlichen Programme nicht wiederhergestellt werden.
Wenn sich vor Aufruf des DOS-Menues ein Programm zur Bedienung einer RS-232-Schnittstelle im Speicher befand und dieses Programm aus irgendwelchen Gründen nicht wieder geladen wird, nachdem Sie mit Hilfe von Funktion B des DOS-Menues ins BASIC zurückgekehrt sind, dann wird der Computer, wenn Sie irgendwann danach die SYSTEM RESET-Taste betätigen, in eine Endlosschleife geraten, aus der Sie nur herauskommen, indem Sie den Computer aus- und wieder einschalten. Das wird jedoch nicht passieren, wenn Sie mit Hilfe der SYSTEM RESET-Taste ins BASIC zurückkehren.

Die Verwendung des DOS-Befehls in einem Programm

Der DOS-Befehl wird hauptsächlich im Eingabemodus verwendet. Er kann zwar auch in Programmen benutzt werden, aber bei seiner Ausführung wird das Programm unterbrochen und es gibt keine Möglichkeit, ein auf diese Weise unterbrochenes Programm fortzusetzen.

DRAWTO (DR.)

Zeichnet eine Linie von dem zuletzt gezeichneten Punkt bis zu dem angebenen Endpunkt.

Format: DRAWTO *Spal, Zeil*
Beispiele: DRAWTO 10,15
DR. SPAL1,ZEIL1
DR. ANFSPAL+SPALOFFSET,ANFZEIL+ZEILOFFSET

Mit Hilfe dieses Befehls wird eine Linie von dem zuletzt mit Hilfe eines PLOT- oder DRAWTO-Befehls gezeichneten Punkt bis zu dem Punkt gezogen, dessen Position durch *Spal* und *Zeil* gekennzeichnet sind. *Spal* und *Zeil* werden gerundet, wenn es sich dabei um nichtganzzahlige Zahlen handelt. Die Linie wird so gerade wie möglich gezeichnet, aber weil es sich immer nur um eine Annäherung an eine ideale Gerade handelt, sieht die Linie meistens ein wenig stufenförmig aus.
Der ATARI-Computer benutzt die Speicherzellen 90 – 92, um die Position des Anfangspunktes für den nächsten DRAWTO-Befehl festzuhalten. In der Speicherzelle 90 befindet sich die Zeilennummer dieses Punktes und in den Speicherzellen 91 und 92 die Nummer der Spalte. Bei Ausführung eines PLOT- oder DRAWTO-Befehls wird der Inhalt dieser Speicherzellen aktualisiert. Alle anderen BASIC-Befehle, wie zum Beispiel GET, PUT oder POSITION haben hierauf keinen Einfluß und verändern somit die Position des Anfangspunktes für den nächsten DRAWTO-Befehl nicht.
Der zuletzt ausgeführte COLOR-Befehl bestimmt, welches Farbregister zur Festlegung der

Zeichenfarbe verwendet wird. Wenn seit dem Einschalten des Computers kein COLOR-Befehl ausgeführt wurde, wird in der Hintergrundfarbe gezeichnet.

Verwendung des DRAWTO-Befehls in den Grafikbetriebsarten 0, 1 und 2

Genauso wie in den Grafikbetriebsarten 3 bis 8, in denen der DRAWTO-Befehl hauptsächlich verwendet wird, kann er auch in den Grafikbetriebsarten 0, 1 und 2 benutzt werden. Die Linie, die dabei von dem zuletzt dargestellten Zeichen bis zu dem Punkt des Bildschirmes gezogen wird, dessen Position durch *Spal* und *Zeil* gekennzeichnet ist, besteht hier allerdings nicht aus Punkten, sondern aus Zeichen. Die Linie wird so gerade wie möglich gezeichnet. Da es sich allerdings immer nur um eine Annäherung an eine ideale Gerade handelt, sieht die Linie meistens etwas stufenförmig aus. Der zuletzt ausgeführte COLOR-Befehl bestimmt, welches Zeichen zum Zeichnen benutzt wird. In den Grafikbetriebsarten 1 und 2 wird durch diesen COLOR-Befehl außerdem die Zeichenfarbe festgelegt (siehe Tabelle 11-3 und 11-4). Wenn seit dem Einschalten des Computers noch kein COLOR-Befehl ausgeführt wurde, wird so verfahren, als wenn ein COLOR 0-Befehl gegeben worden wäre.

END

Beendet ein laufendes Programm.

Format: END
Beispiel: END

Wenn dieser Befehl in einem laufenden Programm auftritt, wird die Programmausführung beendet, die Grafikbetriebsart 0 aktiviert, jegliche Tonerzeugung abgeschaltet und alle Ein/Ausgabekanäle bis auf Kanal 0 geschlossen. Am Ende eines Programmes muß jedoch nicht unbedingt ein END-Befehl stehen. Ein Programm wird automatisch beendet, sobald keine BASIC-Befehle mehr gefunden werden. Es wird dann so verfahren, als wenn ein END-Befehl gegeben worden wäre.

ENTER (E.)

Bei Ausführung des ENTER-Befehls wird ein auf Kassette oder Diskette gespeichertes BASIC-Programm geladen und im Speicher abgelegt.

Format: ENTER *EinGer*
Beispiele: ENTER "C:"
E. PGM$
E. "D2:HAUSHALT.BAS"

Mit Hilfe des ENTER-Befehls wird ein BASIC-Programm von dem Gerät mit dem Gerätenamen *EinGer* geladen. Obwohl dieser Befehl im Prinzip dem CLOAD- oder LOAD-Befehl entspricht, gibt es doch einge sehr wichtige Punkte, in denen er sich von diesen Befehlen unterscheidet.
Bei Ausführung des ENTER-Befehls wird ein eventuell im Speicher des Computers befindliches Programm zum Beispiel nicht gelöscht. Stattdessen wird das neue Programm dem alten überlagert. Falls eine Zeile des neuen Programmes dieselbe Zeilennummer trägt wie eine Zeile des alten Programmes, dann wird die alte Zeile durch die neue ersetzt.
Wenn beim Laden des neuen Programmes ein Variablennamen gefunden wird, der noch

nicht in der Variablentabelle eingetragen ist, dann wird dieser Variablenname zusätzlich eingetragen. Es wird jedoch kein Variablenname, der schon vorher in der Variablentabelle gespeichert war, gelöscht.
Mit Hilfe des ENTER-Befehls können nur BASIC-Programme geladen werden, die im ATASCII-Kode, das heißt also, unter Verwendung des LIST-Befehls, gespeichert worden sind. Programme, die mit Hilfe des SAVE- oder CSAVE-Befehls gespeichert wurden, können so nicht geladen werden.
Zur Übertragung des Programmes vom Programmrecorder oder der Diskettenstation zum Computer wird Kanal 7 verwendet. Der ENTER-Befehl funktioniert auch dann, wenn Kanal 7 bereits belegt war. Allerdings wird der Kanal nach Beendigung der Übertragung geschlossen, so daß er nicht mehr für den Zweck benutzt werden kann, für den er vielleicht vor Ausführung des ENTER-Befehls verwendet wurde. Sobald ein ENTER-Befehl gegeben wird, wird außerdem jegliche Tonerzeugung abgeschaltet.

Verwendung des ENTER-Befehls im Zusammenhang mit dem Programmrecorder

Bei Ausführung des ENTER "C:"-Befehls wird ein auf Kassette gespeichertes Programm mit Hilfe des Programmrecorders geladen und im Speicher abgelegt. Zunächst ertönt ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie nach einer kleinen Pause eine oder mehrere schnelle Tonfolgen hören: Das Programm wird geladen.

Verwendung des ENTER-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des ENTER-Befehls ein bestimmtes File von einer Diskette laden muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN DES DISK OPERATING SYSTEM"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf. Das geschieht auch dann, wenn das Disk Operating System zwar gebootet, aber kein File mit dem angegebenen Filenamen gefunden wurde. Anderenfalls wird das auf Diskette gespeicherte Programm mit dem angegebenen Filenamen geladen und im Speicher abgelegt.

Abbruch des ENTER-Befehls

Die Ausführung eines ENTER-Befehls kann durch Drücken der SYSTEM RESET-Taste unterbrochen werden. Die bis zu diesem Zeitpunkt geladenen Programmzeilen bleiben im Speicher stehen.
Die Ausführung eines ENTER-Befehls wird nicht unterbrochen, wenn Sie die BREAK-Taste drücken. Sie wird nur für einen kurzen Augenblick gestoppt, wobei ein Teil des Programmes, das gerade geladen wird, verloren gehen kann.

Löschen von nicht mehr benötigten Variablennamen

Bei Ausführung eines ENTER-Befehls werden alle Variablennamen des neuen Programmes, die sich noch nicht in der Variablentabelle befinden, zusätzlich eingetragen. Aufgrund dieser Tatsache ist es möglich, Variablennamen, die bei der Ausführung eines Program-

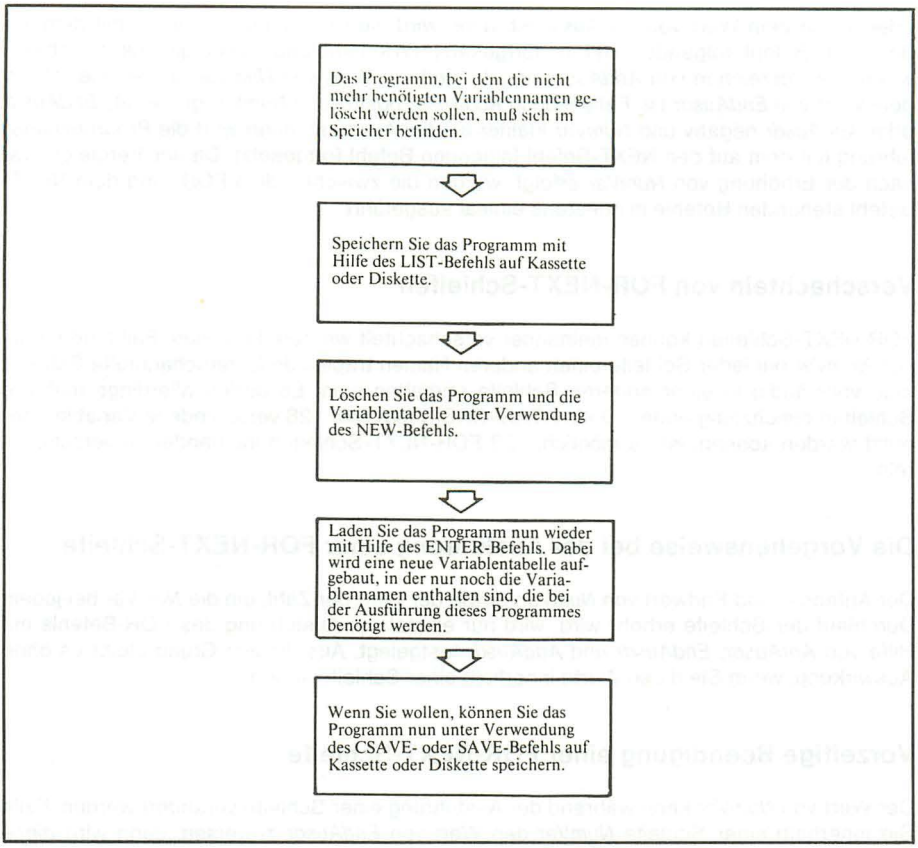


Bild 11-1: Löschen von nicht mehr benötigten Variablennamen

mes nicht benötigt werden, aus der Variablentabelle zu löschen. In Bild 11-1 ist aufgeführt, wie Sie dabei vorgehen müssen.

FOR (F.)

Steht am Anfang einer Serie von Befehlen, die so oft ausgeführt werden, bis eine bei jedem Durchlauf automatisch erhöhte Variable einen bestimmten Wert erreicht hat.

Format: FOR *Numvar* = *AnfAusdr* TO *EndAusdr* [STEP *AddAusdr]*
Beispiele: FOR ZAEHL = 1 TO 100
F. COUNTDOWN = 100 TO 1 STEP -1
F. WERT = ANFANG TO ENDE STEP ERHÖHUNG

Bei der Ausführung des FOR-Befehls wird *NumVar* der Wert von *AnfAusdr* zugewiesen. Danach werden alle darauf folgenden Befehle ausgeführt, bis ein NEXT-Befehl auftritt. Dann wird zu *NumVar* der Wert von *AddAusdr* addiert. Falls der FOR-Befehl ohne den Zusatz 'STEP *AddAusdr*' verwendet wurde, wird 1 addiert. Nun wird der Wert von *NumVar* mit dem Wert von *EndAusdr* verglichen. Die Art des Vergleichs hängt davon ab, welches Vorzeichen *AddAusdr* trägt. Falls dieses Vorzeichen positiv und der Wert von *NumVar* kleiner

oder gleich dem Wert von *EndAusdr* ist, dann wird die Programmausführung mit dem auf den FOR-Befehl folgenden Befehl fortgesetzt (Wiederholung). Das gleiche geschieht, wenn das Vorzeichen von *AddAusdr* negativ und der Wert von *NumVar* größer oder gleich dem Wert von *EndAusdr* ist. Falls jedoch *AddAusdr* positiv und *NumVar* größer als *EndAusdr* oder *AddAusdr* negativ und *NumVar* kleiner als *EndAusdr* ist, dann wird die Programmausführung mit dem auf den NEXT-Befehl folgenden Befehl fortgesetzt. Da der Vergleich erst nach der Erhöhung von *NumVar* erfolgt, werden die zwischen dem FOR- und dem NEXT-Befehl stehenden Befehle mindestens einmal ausgeführt.

Verschachteln von FOR-NEXT-Schleifen

FOR-NEXT-Schleifen können ineinander verschachtelt werden. In diesem Fall muß natürlich *NumVar* bei jeder Schleife einen anderen Namen tragen. Jede verschachtelte Schleife muß vollständig in einer anderen Schleife enthalten sein. Es dürfen allerdings mehrere Schleifen gleichzeitig enden. Da in ATARI-BASIC maximal 128 verschiedene Variablen benutzt werden können, ist es möglich, 128 FOR-NEXT-Schleifen ineinander zu verschachteln.

Die Vorgehensweise bei der Ausführung einer FOR-NEXT-Schleife

Der Anfangs- und Endwert von *NumVar*, sowie der Wert der Zahl, um die *NumVar* bei jedem Durchlauf der Schleife erhöht wird, wird nur einmal bei Ausführung des FOR-Befehls mit Hilfe von *AnfAusdr, EndAusdr* und *AddAusdr* festgelegt. Aus diesem Grund bleibt es ohne Auswirkung, wenn Sie diese Werte innerhalb einer Schleife ändern.

Vorzeitige Beendigung einer FOR-NEXT-Schleife

Der Wert von *NumVar* kann während der Ausführung einer Schleife verändert werden. Falls Sie innerhalb einer Schleife *NumVar* den Wert von *EndAusdr* zuweisen, dann wird diese Schleife beendet, sobald das nächste Mal der NEXT-Befehl erreicht wird.

Benutzung des FOR-Befehls im Eingabemodus

Der FOR-Befehl kann auch im Eingabemodus verwendet werden. Wenn sich jedoch nicht in derselben Zeile, in der der FOR-Befehl auftrat, auch ein NEXT-Befehl befindet, wird die Schleife nur einmal ausgeführt.

Was bei der Benutzung von FOR-NEXT-Schleifen zu beachten ist

Springen Sie nicht, nachdem Sie eine Schleife im Hauptprogramm begonnen haben, in ein Unterprogramm. Am Anfang einer Schleife muß immer ein FOR-Befehl stehen; springen Sie deshalb nicht in eine FOR-NEXT-Schleife. Springen Sie auch nicht aus einer FOR-NEXT-Schleife, weil der Computer dann immer noch einen NEXT-Befehl erwartet, was eventuell sogar zu einem Fehler führen kann.

GET (GE.)

Lädt unter Verwendung eines Ein/Ausgabekanals ein einzelnes Byte von dem mit diesem Ein/Ausgabekanal angesprochenen Gerät.

Format: GET #*Kan, NumVar*
Beispiele: *GET* #1,ZAHL
GET #KA,X

Kanal *Kan* muß für Eingabe von einem bestimmten Gerät geöffnet sein. Bei Ausführung des GET-Befehls wird von diesem Gerät ein einzelnes Byte geladen und in Form einer Zahl, deren Wert zwischen 0 und 255 liegt, der Variablen *NumVar* zugewiesen.

Verwendung des GET-Befehls im Zusammenhang mit der Tastatur

Wenn Kanal *Kan* für Eingabe von der Tastatur (Gerätename K:) geöffnet ist, wird bei Ausführung eines GET-Befehls *NumVar* der Dezimalwert des ATASCII-Kodes der nächsten Taste oder Tastenkombination, die gedrückt wird, zugewiesen. In Teil D des Anhangs sind diese Werte aufgeführt. Die Programmausführung wird bis zum Auftreten des nächsten Tastendrucks gestoppt.
Beim Drücken der BREAK-Taste wird kein ATASCII-Kode erzeugt, sondern die Ausführung des GET-Befehls unterbrochen. Beim Drücken von CTRL-3 tritt ein Fehler auf und wenn Sie CTRL-1 drücken, wird, wie üblich, die Bildschirmausgabe unterbrochen. Beim Drücken der ⅄ , CAPS/LOWR-, SHIFT- und CTRL-Taste wird kein ATASCII-Kode erzeugt. Mit Hilfe dieser Tasten können jedoch die Kodes der anderen Tasten verändert werden. Die vier gelben Tasten, die normalerweise für Sonderfunktionen verwendet werden, erzeugen ebenfalls keine ATASCII-Kodes.

Verwendung des GET-Befehls im Zusammenhang mit dem Programmrecorder

Die Datenübertragung vom Programmrecorder zum Computer geschieht in Blöcken zu je 128 Bytes. Wenn Kanal *Kan* für Eingabe von dem Programmrecorder geöffnet ist, liest der Computer, wenn zum ersten Mal ein GET-Befehl ausgeführt wird, einen ganzen Block, speichert diesen Block im Kassettenbuffer, weist das erste Byte des Blockes der Variablen *NumVar* zu und stoppt den Programmrecorder. Falls nun weitere GET-Befehle folgen, wird ein Byte des Blockes nach dem anderen aus dem Kassettenbuffer gelesen. Wenn der ganze Block gelesen worden ist, läßt der Computer den Programmrecorder beim Auftreten des nächsten GET-Befehls wieder anlaufen und liest den nächsten Block.
Falls Sie versuchen sollten, über das Ende eines Files hinaus weitere Daten zu lesen, tritt ein Fehler auf. Beim Schließen des Eingabekanals mit Hilfe eines CLOSE- oder END-Befehls wird der Programmrecorder gestoppt.

Verwendung des GET-Befehls im Zusammenhang mit der Diskettenstation

Es werden Daten von dem File gelesen, das für Eingabe geöffnet worden ist. Mit Hilfe des GET-Befehls können sowohl Daten, die unter Verwendung des PUT-Befehls gespeichert wurden und aus einem Byte bestehen, als auch Daten, die unter Verwendung des PRINT-Befehls gespeichert wurden und aus mehreren Bytes bestehen, gelesen werden. Die unter Verwendung des PRINT-Befehls gespeicherten Daten werden dabei natürlich Byte für Byte gelesen.
Die Datenübertragung von der Diskettenstation zum Computer geschieht nicht zeichenweise, sondern in Blöcken. Diese Blöcke sind genauso lang wie ein Sektor der Diskette. Sobald ein File für Eingabe geöffnet wird, liest der Computer den ersten Block und legt ihn im Diskettenbuffer ab. Bei der Ausführung des ersten darauffolgenden GET-Befehls wird das erste Byte dieses Blockes der Variablen *NumVar* zugewiesen, bei der Ausführung des

zweiten GET-Befehls das zweite Byte und so weiter. Nachdem der ganze Block gelesen worden ist, lädt der Computer den nächsten Sektor von der Diskette. Bei der ATARI-Diskettenstation 810 enthält jeder Sektor 125 Bytes.
Auch wenn bei der Ausführung eines POINT-Befehls ein anderer Sektor angesprochen wird als der, der sich gerade im Diskettenbuffer befindet, lädt der Computer einen neuen Block von der Diskette.

Verwendung des GET-Befehls im Zusammenhang mit dem Bildschirm

Wenn Kanal *Kan* für Eingabe von dem Bildschirm (Gerätename S: oder E:) geöffnet ist, wird bei Ausführung eines GET-Befehls der Kode des Zeichens oder Punktes gelesen, das sich an der momentanen Cursorposition befindet. Diese Kodezahl entspricht der Zahl, die bei einem COLOR-Befehl angegeben werden mußte, um dieses Zeichen beziehungsweise diesen Punkt auf dem Bildschirm darzustellen. In der Grafikbetriebsart 0 kann mit Hilfe dieser Kodezahl festgestellt werden, welches Zeichen sich an der momentanen Cursorposition befindet (siehe Tabelle 11-3). In den Grafikbetriebsarten 1 und 2 kann außer dem Zeichen selbst auch die Zeichenfarbe bestimmt werden (siehe Tabelle 11-4). In den Grafikbetriebsarten 3 bis 8 enthält die Kodezahl eine Information darüber, welches Farbregister die Farbe des an der momentanen Cursorposition befindlichen Punktes bestimmt (siehe Tabelle 11-1).
Nach der Benutzung des GET-Befehls im Zusammenhang mit dem Bildschirm wird die Position des Cursors um eine Stelle nach rechts verschoben. Dabei wird in der Speicherzelle 84 die Zeilennummer und in den Speicherzellen 85 und 86 die Spaltennummer der neuen Cursorposition gespeichert. Der nächste Befehl, mit dessen Hilfe Daten auf den Bildschirm gebracht oder von dem Bildschirm gelesen werden, bezieht sich auf diese Cursorposition. Dies gilt jedoch nicht für den DRAWTO- und die XIO-Befehle, weil diese Befehle die in den Speicherzellen 90-92 gespeicherte Cursorposition verwenden.
Wenn sich der Cursor in der letzten Spalte einer Zeile befindet, wird er nach der Ausführung eines GET-Befehls in die erste Spalte der nächsten Zeile verschoben. Falls sich der Cursor in der letzten Spalte der letzten Zeile befand, muß er nach der Ausführung eines GET-Befehls neu positioniert werden. Anderenfalls wird ein Fehler auftreten, sobald das nächste Mal auf den Bildschirm zugegriffen wird. Wenn nach einem GET-Befehl ein PRINT-Befehl ausgeführt wird, kann es vorkommen, daß das Zeichen oder der Punkt, der gerade gelesen wurde, verändert wird. Um das zu verhindern, verschieben Sie den Cursor nach der Ausführung eines GET-Befehls mit Hilfe des POSITION-Befehls wieder an die Stelle, an der er sich vor Ausführung des GET-Befehls befand und schreiben unter Verwendung des PUT-Befehls die Kodezahl auf den Bildschirm, die Sie gerade vorher gelesen haben.

Verwendung des GET-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Kanal *Kan* muß für Eingabe von der gewünschten RS-232-Schnittstelle des ATARI-Interfacemoduls 850 geöffnet sein. Das funktioniert nur dann, wenn ein Programm zur Bedienung der RS-232-Schnittstellen beim Einschalten des Computers gebootet wurde. Außerdem muß die serielle Schnittstelle mit einem XIO 40-Befehl für gleichzeitige Ein- und Ausgabe eingestellt und mit Hilfe eines XIO 38-Befehls festgelegt werden, ob und wie die empfangenen Daten übersetzt werden sollen. Bevor diese Aufgaben nicht erledigt sind, ist es nicht möglich, mit Hilfe des GET-Befehls Daten von einer seriellen Schnittstelle zu lesen.
Wenn nun von einem externen Gerät Daten zum ATARI-Interfacemodul 850 übertragen werden, dann werden diese Daten entsprechend der eingestellten Betriebsart der Schnittstelle in ATASCII-Zeichen übersetzt. In Teil D des Anhanges sind die Kodezahlen des ATASCII- und des ASCII-Kodes einander gegenübergestellt. Danach werden die empfangenen Daten Byte für Byte an den Computer übergeben.

GOSUB (GOS.)

Beim Auftreten eines GOSUB-Befehls wird das Programm mit der Zeile, die die angegebene Zeilennummer trägt, fortgesetzt. Sobald danach im weiteren Verlauf des Programmes ein RETURN-Befehl auftritt, erfolgt ein Rücksprung zu der direkt auf den GOSUB-Befehl folgenden Anweisung.

Format: GOSUB *ProZeil*
Beispiele: *GOSUB 100*
GOS. PHYTRECHN
GOS. ANFANG+ZUSATZ

Mit Hilfe des GOSUB-Befehls wird ein Unterprogramm aufgerufen. Die Ausführung dieses Unterprogrammes beginnt mit der Zeile, die die Zeilennummer *ProZeil* trägt. Das muß nicht unbedingt die erste Zeile des Unterprogrammes sein.
Falls es keine Zeile mit der angegebenen Zeilennummer gibt, tritt ein Fehler auf.

Beendigung eines Unterprogrammes

Bei der Ausführung eines GOSUB-Befehls wird die Adresse, zu der nach Beendigung des Unterprogrammes zurückgesprungen werden soll, in einem speziellen Speicherbereich, dem sogenannten Stack, gespeichert. Diese Adresse bezieht sich auf den BASIC-Befehl, der unmittelbar auf den GOSUB-Befehl folgt. Er kann auch in derselben Programmzeile wie der GOSUB-Befehl selbst stehen. Sobald nach dem Aufruf des Unterprogrammes ein RETURN-Befehl auftritt, wird die Programmausführung mit diesem Befehl fortgesetzt und gleichzeitig die Rücksprungadresse aus dem Stack gelöscht. Damit ist das Unterprogramm beendet. Wenn Sie jedoch zum Beispiel mit Hilfe eines GOTO-Befehls ein Unterprogramm verlassen, dann wird die Rücksprungadresse nicht gelöscht, sondern bleibt im Stack stehen und belegt dort unnötig Speicherplatz. Wenn so etwas in einem Programm mehrmals vorkommt, wird der für den Stack zur Verfügung stehende Speicherplatz verbraucht und es tritt ein Fehler auf. Sie können zwar mit Hilfe eines GOTO-, IF-THEN- oder eines ähnlichen Befehls aus einem Unterprogramm springen, aber dann müssen Sie vorher die Rücksprungadresse unter Verwendung des POP-Befehls löschen.
Auch wenn Sie das beachten, kann es in seltenen Fällen Probleme mit dem Stack geben, weil dafür nur ein begrenzter Speicherbereich zur Verfügung steht und deshalb nicht beliebig viele Unterprogramme ineinander verschachtelt werden können. Auch bei der Ausführung von FOR-NEXT-Schleifen wird eine Rücksprungadresse im Stack gespeichert. Die Anzahl der Unterprogramme, die ineinander verschachtelt werden können, hängt also davon ab, wieviele ineinander verschachtelte FOR-NEXT-Schleifen zur gleichen Zeit benutzt werden.

Die Plazierung eines Unterprogrammes

Ein Unterprogramm kann an jeder beliebigen Stelle eines Programmes plaziert werden. Es muß jedoch immer mit dem Anfang einer Zeile beginnen.

GOTO (G.)

Beim Auftreten eines GOTO-Befehls wird die Programmausführung mit der Zeile fortgesetzt, die die Zeilennummer *ProZeil* trägt.

Format: *GOTO ProZeil*
Beispiele: *GOTO 1120*
G. TABELLE+OFFSET

Nach der Ausführung eines GOTO-Befehls wird die Programmausführung sofort mit dem ersten Befehl der angegebenen Programmzeile fortgesetzt. Falls es keine Zeile mit der angegebenen Zeilennummer gibt, tritt ein Fehler auf.

GRAPHICS (GR.)

Aktiviert eine der Grafikbetriebsarten.

Format: GRAPHICS *NumAusdr*
Beispiele: GRAPHICS 5
GRAPHICS 20
GR. 32+BETART

Bei Ausführung eines GRAPHICS-Befehls wird die durch *NumAusdr* gekennzeichnete Grafikbetriebsart aktiviert. *NumAusdr* wird gerundet, wenn es sich dabei um eine nichtganzzahlige Zahl handelt. Normalerweise wird bei der Ausführung eines GRAPHICS-Befehls der Bildschirm gelöscht. Dies kann jedoch verhindert werden, indem Sie zu der Nummer der gewünschten Grafikbetriebsart 32 hinzuzählen und diese Zahl für *NumAusdr* einsetzen. In Tabelle 11-6 ist aufgeführt, welche Werte für *NumAusdr* eingesetzt werden können, um eine bestimmte Grafikbetriebsart zu aktivieren.
Bei der Ausführung eines GRAPHICS-Befehls reserviert der Computer entsprechend der gewählten Grafikbetriebsart einen gewissen Speicherbereich für den Bildschirmspeicher, aktiviert den Cursor, indem er in der Speicherzelle 752 eine 0 ablegt (betrifft nur das Textfenster) und ändert die Werte der Farbregister entsprechend den Voreinstellungen (siehe Tabelle 11-2).

Tabelle 11-6: Zusammenstellung der Werte, die beim GRAPHICS *NumAusdr*-Befehl für *NumAusdr* eingesetzt werden können

Grafikbetriebsart	Dasselbe ohne Textfenster	Dasselbe, ohne den Bildschirm zu löschen	Dasselbe, ohne Textfenster und ohne den Bildschirm zu löschen
0	–	32	–
1	17	33	49
2	18	34	50
3	19	35	51
4	20	36	52
5	21	37	53
6	22	38	54
7	23	39	55
8	24	40	56

Die Grafikbetriebsarten

In ATARI-BASIC gibt es neun verschiedene Grafikbetriebsarten. In der Grafikbetriebsart 0 wird nur Text dargestellt. Diese Grafikbetriebsart wird auch beim Einschalten des Computers aktiviert. In den Grafikbetriebsarten 1 bis 8 befindet sich normalerweise am unteren Bildschirmrand ein Textfenster, in dem vier Zeilen normaler Text (wie in der Grafikbetriebs-

art 0) dargestellt werden können. Dieses Textfenster kann jedoch auch ausgeblendet werden. In Tabelle 11-7 sind die charakteristischen Eigenschaften aller Grafikbetriebsarten aufgeführt.

Tabelle 11-7: Die charakteristischen Eigenschaften aller Grafikbetriebsarten

Grafik-betriebs-art	Anzahl der Zeilen		Anzahl der Spalten	Benutzte Farbregister	Benötigter Speicher-platz (in Bytes)	Verwendung
	Ohne Text-fenster	Mit Text-fenster				
0	24	–	40	1*, 2†, 4#	993	Text
1	24	20	20	0, 1, 2, 3, 4†#	513	Blocktext und Grafik
2	12	10	20	0, 1, 2, 4, 4†#	261	Blocktext und Grafik
3	24	20	40	0, 1, 2, 4 †#	273	Grafik
4	48	40	80	0, 4†#	537	Grafik
5	48	40	80	0, 1, 2, 4†#	1017	Grafik
6	96	80	160	0, 4†#	2025	Grafik
7	96	80	160	0, 1, 2, 4†#	3945	Grafik
8	192	160	320	1*, 2†, 4#	7900	Grafik

* Bestimmt nur die Helligkeit des Textes oder der Grafik. Die Farbe entspricht der des Hintergrundes.
† Bestimmt die Farbe und die Helligkeit des Hintergrundes.
# Bestimmt die Farbe und die Helligkeit der Umrandung

Das Textfenster

In den Grafikbetriebsarten 1 bis 8 befindet sich am unteren Bildschirmrand ein Textfesnter, in dem vier Zeilen normaler Text dargestellt werden können. Über Kanal 0 ausgegebener Text wird in dieses Textfenster geschrieben.
Wenn Sie bei der Ausführung des GRAPHICS-Befehls zur Nummer der gewünschten Grafikbetriebsart 16 hinzuzählen und diesen Wert für *NumAusdr* einsetzen (siehe Tabelle 11-6), dann wird das Textfenster nicht eingeblendet. Der Bereich des Bildschirmes, der sonst dafür in Anspruch genommen wird, steht dann ebenfalls für die gewählte Grafikbetriebsart zur Verfügung.
Wenn das Textfenster nicht eingeblendet ist, kann natürlich auch kein Text in das Textfenster geschrieben werden. Dazu zählt auch das Fragezeichen, das bei Ausführung eines INPUT-Befehls und der Text, der über Kanal 0 oder bei Ausführung eines normalen PRINT-Befehls ausgegeben wird und die Meldung, die bei einer durch einen Fehler oder einen STOP-Befehl oder durch Drücken der BREAK-Taste verursachten Programmunterbrechung auf dem Bildschirm erscheint. Wenn in den Grafikbetriebsarten 1 bis 8 eine dieser Ausgaben erfolgt und kein Textfenster eingeblendet ist, dann wird der ganze Bildschirm gelöscht, auf die Grafikbetriebsart 0 umgeschaltet und der entsprechende Text am oberen Ende des Bildschirmes ausgegeben.

Ausgabe über Kanal 0 und Kanal 6

Bei der Ausführung eines GRAPHICS-Befehls wird Kanal 6 für Ausgabe auf dem Bildschirm (Gerätename S:) geöffnet. Danach können Sie diesen Kanal nur dann anderweitig benutzen, wenn Sie ihn vorher mit Hilfe eines CLOSE #6-Befehls geschlossen haben. Dann ist allerdings kein DRAWTO-, PLOT- oder LOCATE-Befehl mehr ausführbar, bevor dieser Kanal nicht wieder unter Verwendung eines GRAPHICS- oder OPEN-Befehls für Ausgabe auf dem Bildschirm geöffnet worden ist.

Bei der Ausführung eines GRAPHICS-Befehls wird aber nicht nur Kanal 6 für Ausgabe auf dem Bildschirm, sondern auch Kanal 0 für Ausgabe auf das Textfenster geöffnet. Diese Ausgabe erfolgt über den Bildschirmeditor (Gerätename E:). In der Grafikbetriebsart 0 wird auf dem ganzen Bildschirm Text dargestellt. Hier ist es absolut gleichwertig, ob eine Ausgabe über Kanal 0 oder Kanal 6 erfolgt. In den Grafikbetriebsarten 1 bis 8 ist das nicht der Fall. Kanal 0 ist hier für das Textfenster und Kanal 6 für den übrigen Bildschirm zuständig. Falls kein Textfenster eingeblendet ist, wird, sobald eine Ausgabe über Kanal 0 erfolgt, wieder auf die Grafikbetriebsart 0 umgeschaltet.

Die Grafikbetriebsart 0

In der Grafikbetriebsart 0 wird ausschließlich Text dargestellt. Der Bildschirm ist in 24 Zeilen zu je 40 Zeichen unterteilt. Die ersten zwei Spalten von jeder Zeile können jedoch normalerweise nicht beschrieben werden, weil das durch die linke Bildschirmbegrenzung verhindert wird.
Die Farbe der in der Grafikbetriebsart 0 dargestellten Zeichen entspricht der des Hintergrundes. Es ist jedoch möglich, die Helligkeit der Zeichen zu verändern (siehe Tab. 11-7).

Die in der Grafikbetriebsart 0 geltenden Bildschirmbegrenzungen

Mit Hilfe eines POKE-Befehls können Sie die rechte und die linke Bildschirmbegrenzung frei wählen. Der für eine Begrenzung eingesetzte Wert muß zwischen 0 und 39 liegen. Der Inhalt der Speicherzelle 82 bestimmt die linke und der Inhalt der Speicherzelle 83 die rechte Bildschirmgrenze. Alles, was mit Hilfe eines PRINT-Befehls auf den Bildschirm geschrieben wird, wird innerhalb dieser Grenzen dargestellt, aber Sie können trotzdem noch auf den ganzen Bildschirm zugreifen.

Die Funktion des Cursors in der Grafikbetriebsart 0

Der Cursor steht dort, wo das nächste Zeichen ausgegeben wird. Die Position des Cursors kann unter Verwendung des POSITION-Befehls verändert werden. Mit Hilfe der Speicherzellen 752 und 755 wird die Funktion des Cursors beeinflußt (siehe Teil G des Anhanges).

Der Zeichenvorrat der Grafikbetriebsart 0

In der Grafikbetriebsart 0 gibt es 128 verschiedene Zeichen. Jedes Zeichen kann normal oder invertiert dargestellt werden. Welches Zeichen des Standardzeichensatzes welche Kodezahl trägt, können Sie Tabelle 11-3 entnehmen. Eine Möglichkeit, wie Sie sich Ihren eigenen Zeichensatz definieren können, wurde in Kapitel 9 beschrieben.

Definition der logischen Zeile

Der auf dem Bildschirm dargestellte Text wird in der Grafikbetriebsart 0 in logische Zeilen unterteilt. Eine logische Zeile kann 1 bis 120 Zeichen oder maximal drei Bildschirmzeilen enthalten. Das Ende einer logischen Zeile ist durch ein End-of-line-Zeichen (EOL) gekennzeichnet.
Wenn der Cursor den unteren Bildschirmrand erreicht, dann wird der ganze Bildschirminhalt nach oben verschoben, so daß die am oberen Bildschirmrand dargestellte logische Zeile verschwindet und am unteren Bildschirmrand weitergeschrieben werden kann.

Die Grafikbetriebsarten 1 und 2

In den Grafikbetriebsarten 1 und 2 kann Blocktext oder Grafik dargestellt werden. Der hierbei geltende Zeichensatz umfaßt 64 Zeichen. Die in der Grafikbetriebsart 2 dargestellten Zeichen sind genauso breit, aber doppelt so hoch wie die Zeichen, die in der Grafikbetriebsart 1 dargestellt werden. In beiden Fällen sind die Zeichen doppelt so breit wie die Zeichen der Grafikbetriebsart 0 (siehe Tabelle 11-7).
In den Grafikbetriebsarten 1 und 2 kann zwischen zwei verschiedenen Zeichensätzen gewählt werden. Der erste Zeichensatz enthält Großbuchstaben, Zahlen und Satzzeichen und der zweite Kleinbuchstaben und Grafikzeichen. Nach der Ausführung eines GRAPHICS-Befehls ist immer der erste Zeichensatz aktiviert. Mit Hilfe des Befehls POKE 756,226 wird auf den zweiten und mit POKE 756,224 wieder auf den ersten Zeichensatz umgeschaltet. In Tabelle 11-4 sind die Zeichen von beiden Zeichensätzen aufgeführt
Die Zeichen können in vier verschiedenen Farben dargestellt werden. Das fünfte Farbregister bestimmt die Farbe des Hintergrundes und der Umrandung. Der Kode, der zur Darstellung eines bestimmten Zeichens verwendet wird, bestimmt gleichzeitig auch, welches Farbregister bei der Darstellung dieses Zeichens verwendet wird (siehe Tabelle 11-4). Die Farben selbst werden, wie üblich, mit Hilfe des SETCOLOR-Befehls festgelegt.
Wenn Sie mehr Zeichen ausgeben, als in einer Zeile Platz finden, dann wird eine neue Zeile begonnen. In den Grafikbetriebsarten 1 und 2 wird der Bildschirminhalt nicht nach oben verschoben, wenn der untere Bildschirmrand erreicht worden ist. Wenn Sie in diesem Fall noch weitere Zeichen an den Bildschirm ausgeben, tritt ein Fehler auf.

Die Grafikbetriebsarten 3 bis 8

In den Grafikbetriebsarten 3 bis 8 wird Grafik dargestellt. Die Anzahl der Zeilen, die Anzahl der Punkte pro Zeile und die Größe der Punkte ist bei jeder Grafikbetriebsart verschieden. In Tabelle 11-7 sind die charakteristischen Eigenschaften von jeder Grafikbetriebsart aufgeführt.
Obwohl der Cursor nicht sichtbar ist, kann er an jede beliebige Stelle des Bildschirmes bewegt werden. Bei Ausführung eines POSITION-Befehls wird die in der Speicherzelle 84 gespeicherte Zeilennummer und die in den Speicherzellen 85 und 86 gespeicherte Spaltennummer entsprechend verändert. In diesen Speicherzellen ist nicht die momentane Cursorposition, sondern die Position gespeichert, an die der Cursor bewegt wird, wenn das nächste Mal der Bildschirm angesprochen wird. Die momentane Cursorposition ist in den Speicherzellen 90 (Zeilennummer), 91 und 92 (Spaltennummer) festgehalten. Sie wird bei Ausführung eines DRAWTO-, PLOT- oder XIO-Befehls benötigt.
Der DRAWTO-, PLOT- und XIO-Befehl sind die Befehle, die in den Grafikbetriebsarten 3 bis 8 hauptsächlich verwendet werden, aber natürlich können Sie alle anderen Befehle auch benutzen.

IF-THEN

Wenn eine bestimmte Bedingung erfüllt ist, werden der oder die angegebenen Befehle ausgeführt.

Format: IF *Ausdr* THEN *Befehl* [:*Befehl*...]
IF *Ausdr* THEN *ProZeil*

Beispiele:
```
IF NAME$="KNUT BUDDNASE" THEN RETURN
IF PLZ>8000 AND NAME$(1,1)<="B" THEN PRINT #2;NAME$
IF ANTWORT$="J" THEN PRINT "WIEVIEL";:INPUT ANZ
IF A=B THEN 1735
IF VAR(N1,N2) THEN 25300+VAR(N1,N2)/1E4
```

Falls Sie das erste oben angegebene Format benutzen, werden die auf 'THEN' folgenden Befehle derselben Programmzeile ausgeführt, wenn die mit Hilfe von *Ausdr* angegebene Bedingung erfüllt ist. Wenn die Bedingung nicht erfüllt ist, wird keiner der auf 'THEN' folgenden Befehle ausgeführt, sondern die Programmausführung mit dem ersten Befehl der nächsten Programmzeile fortgesetzt. Falls Sie das zweite oben angegebene Format benutzen, wird das Programm mit der angegebenen Programmzeile fortgesetzt, wenn die mit Hilfe von *Ausdr* angegebene Bedingung erfüllt ist. Anderenfalls wird auch hier die Programmausführung mit dem ersten Befehl der nächsten Zeile fortgesetzt.

Die verschiedenen Arten von Bedingungen, die für *Ausdr* eingesetzt werden können

Meistens wird für *Ausdr* ein Vergleich oder ein boolescher Ausdruck eingesetzt. Auch ein numerischer Ausdruck kann verwendet werden. Wenn sein Wert ungleich 0 ist, dann gilt die Bedingung als erfüllt. Wenn sein Wert gleich 0 ist, dann gilt die Bedingung als nicht erfüllt und die Programmausführung wird mit dem ersten Befehl der Zeile fortgesetzt, die direkt auf die Zeile folgt, in der sich der IF-THEN-Befehl befindet. Bei *Ausdr* darf es sich nicht um einen alphanumerischen Ausdruck handeln, aber es können sehr wohl mit Hilfe von *Ausdr* alphanumerische Ausdrücke miteinander verglichen werden.
Wenn es sich bei *Ausdr* um einen Vergleich mit Hilfe des Größer- (>), Kleiner- (<) oder des Ungleich-Zeichens (< >) handelt, dann darf nur auf einer Seite dieses Zeichens eine STR$-Funktion stehen. Dasselbe gilt auch für die CHR$-Funktion.

Vorgehensweise beim Vergleichen von alphanumerischen Ausdrücken

Wenn es sich bei *Ausdr* um einen Vergleich von alphanumerischen Ausdrücken handelt, dann werden diese Ausdrücke Zeichen für Zeichen miteinander verglichen und zwar mit Hilfe der ATASCII-Kodes der Zeichen (siehe Teil D des Anhanges). Dieser Vergleich erfolgt so lange, bis sich zwei Zeichen voneinander unterscheiden. Es gilt dann der alphanumerische Ausdruck als größer, in dem das Zeichen mit dem höheren ATASCII-Kode enthalten ist. Falls kein Unterschied beim Vergleich festgestellt wird, ist der längere alphanumerische Ausdruck der größere.

Was bei der Benutzung des IF-THEN-Befehls zu beachten ist

Falls es sich bei einem von den auf 'THEN' folgenden Befehlen um einen REM- oder GOTO-Befehl handelt, dann muß dieser Befehl als letzter in der Programmzeile stehen, weil darauf folgende Befehle niemals ausgeführt würden.
Auf ein 'THEN' kann ohne weiteres ein anderer IF-THEN-Befehl folgen, solange dieser Befehl vollständig in derselben Zeile wie der erste IF-THEN-Befehl enthalten ist. Ein boolescher Ausdruck ist jedoch erheblich leichter zu lesen als auf diese Weise verschachtelte IF-THEN-Befehle. Bei der Ausführung der beiden folgenden Befehle geschieht zum Beispiel exakt dasselbe; nur ist der zweite Befehl leichter zu lesen und zu verstehen:

```
10 IF A$="X" THEN IF B=2 THEN IF C>D THEN 50
10 IF A$="X" AND B=2 AND C>D THEN 50
```

INPUT (I.)

Bei Ausführung des INPUT-Befehls werden von dem angesprochenen Gerät Zeichen empfangen, umgeformt und in Form von numerischen oder alphanumerischen Werten der oder den Variablen zugewiesen.

Format: INPUT [#*Kan* {;/,}] *Var* [,*Var* ...]

Beispiele: INPUT ANTWORT
I. #4,RECORD$
I. #2,A,B,C

Bei Ausführung eines INPUT-Befehls werden von einem bestimmten Gerät Daten empfangen. Diese Daten bestehen aus einem oder mehreren ATASCII-Zeichen, gefolgt von einem End-of-line-Zeichen (EOL) oder auch nur aus einem EOL-Zeichen. Bei der Tastatur wird beim Drücken der RETURN-Taste ein EOL-Zeichen erzeugt. Aber unabhängig davon, von welchem Gerät Daten empfangen werden: es muß auf jeden Fall ein EOL-Zeichen auftreten, um die Ausführung des INPUT-Befehls zu beenden.
Ob der Computer die eingelesenen Daten als numerische oder alphanumerische Werte oder eine Kombination von beiden interpretiert, hängt davon ab, wieviele und welche Art von Variablen *(Var)* angegeben werden. Es spielt keine Rolle, von welchem Gerät die Daten empfangen werden.
Wenn keine Kanalnummer angegeben wird, werden mit Hilfe des Bildschirmeditors (Gerätename E:) Daten von der Tastatur empfangen. Falls eine Kanalnummer angegeben wird, muß vorher der Kanal *Kan* unter Verwendung eines OPEN-Befehls für Eingabe von dem gewünschten Gerät geöffnet worden sein.

Einlesen von mehreren Werten auf einmal

Wenn mit Hilfe eines einzigen INPUT-Befehls mehrere Werte auf einmal eingelesen werden sollen, dann müssen diese Werte, ganz gleich, ob es sich nun um numerische oder alphanumerische Werte handelt, bei der Eingabe durch EOL-Zeichen voneinander getrennt sein, was bei der Tastatur durch Drücken der RETURN-Taste erreicht wird. Numerische Werte können aber auch durch Kommas voneinander getrennt werden. In diesem Fall brauchen keine EOL-Zeichen eingefügt werden. Das funktioniert jedoch nicht bei alphanumerischen Werten, weil dort das Komma als ein Teil des alphanumerischen Wertes angesehen werden würde.

Einlesen von Zahlen

Wenn eine Zahl eingelesen werden soll, dann empfängt das BASIC solange Daten, bis ein Komma oder ein EOL-Zeichen auftritt, formt sie entsprechend um und weist den auf diese Weise entstandenen numerischen Wert der angegebenen numerischen Variablen zu. Die eingelesenen Daten müssen genauso aufgebaut sein wie eine numerische Konstante (siehe Kapitel 3). Zuerst kommt ein Vorzeichen (+ oder –). Dieses Vorzeichen muß jedoch nicht unbedingt vorhanden sein. Darauf folgen dann eine oder mehrere Ziffern, zwischen denen ein Dezimalpunkt stehen kann. Am Anfang und am Ende der Zahl dürfen Leerzeichen stehen, aber das Vorzeichen, die Ziffern und der Dezimalpunkt dürfen nicht durch Leerzeichen voneinander getrennt werden.
Bei Verwendung der Exponentialschreibweise folgt nun zur Angabe des Exponenten noch ein Zusatz. Dieser Zusatz besteht aus drei Teilen: dem Buchstaben E, einem Vorzeichen, das jedoch auch hier nicht unbedingt vorhanden sein muß und einer oder zwei Ziffern. Der Wert des Exponenten darf zwischen -99 und -1 beziehungsweise zwischen 1 und 97 liegen. Nicht verwendet werden dürfen Brüche oder die 0. Eine Zahl besteht aber nicht nur aus Exponenten, sondern auch aus einer Mantisse (die Zahl, die vor dem Exponenten steht). Der Wert dieser Zahl muß innerhalb bestimmter Grenzen liegen (siehe unten). Zwischen der Mantisse und dem Exponenten dürfen keine Leerzeichen stehen.
Der Wert der eingelesenen Zahlen muß größer sein als –1E+98 und kleiner als 1E+98. Für

eine Zahl, deren Absolutwert kleiner als 9.99999999E-98 ist, wird 0 eingesetzt.
Wenn beim Einlesen einer Zahl sofort ein Komma oder ein EOL-Zeichen festgestellt wird, tritt ein Fehler auf. Bei der Tastatur geschieht das, wenn nur die RETURN-Taste gedrückt wird. Auch wenn ein anderes Zeichen als die Ziffern von 0 bis 9, +, –, E oder ein Punkt gelesen wird, tritt ein Fehler auf. Dasselbe geschieht, wenn mehrere Dezimalpunkte vorhanden sind, das Vorzeichen an der falschen Stelle steht oder der Wert des Exponenten zu groß oder zu klein ist.

Einlesen von alphanumerischen Werten

Am Ende eines alphanumerischen Wertes muß grundsätzlich ein EOL-Zeichen stehen, weil Kommas als Teil des alphanumerischen Wertes gelten. Bei der Tastatur wird das durch Drücken der RETURN-Taste erreicht. Die gelesenen Daten werden direkt ohne irgendeine Umformung der angegebenen alphanumerischen Variablen zugewiesen. Wenn gleich als erstes ein EOL-Zeichen gelesen wird, dann ist die Länge der Variablen gleich 0. Es kann natürlich auch einmal vorkommen, daß bis zum Auftreten eines EOL-Zeichens mehr Daten gelesen werden, als der angegebenen alphanumerischen Variablen aufgrund ihrer Dimensionierung zugewiesen werden können. In diesem Fall werden die Daten, für die kein Platz mehr vorhanden ist, ignoriert.

Verwendung des INPUT-Befehls im Zusammenhang mit dem Editor oder der Tastatur

Wenn zum Einlesen von Daten mit Hilfe des INPUT-Befehls der Editor verwendet wird (Gerätename E:), dann wird in der Grafikbetriebsart 0 an der momentanen Cursorposition ein Fragezeichen dargestellt und damit angezeigt, daß nun eine Eingabe erfolgen muß. Dies geschieht jedoch nicht, wenn eine Kanalnummer angegeben wurde oder die Eingabe nicht über den Editor läuft, also auch dann nicht, wenn die Eingabe direkt über die Tastatur (Gerätename K:) läuft. Wenn mit Hilfe eines INPUT-Befehls mehrere Werte auf einmal gelesen werden sollen und dabei mehrere Zeilen benutzt werden (siehe oben), dann erscheint am Anfang von jeder Folgezeile erneut ein Fragezeichen. Damit wird angezeigt, daß noch weitere Eingaben folgen.
Wenn Sie die Tastatur (Gerätename E: oder K:) mit einem INPUT-Befehl ansprechen, ändert sich dadurch die Funktion der Tastatur nicht. Der Cursor kann wie gewohnt auf dem Bildschirm bewegt werden, beim Drücken der SHIFT- und der CLEAR-Taste wird der Bildschirm gelöscht, bei Betätigung der BREAK-Taste wird die Ausführung des INPUT-Befehls unterbrochen, beim Drücken der RETURN-Taste wird die Eingabe beendet und so weiter und so weiter. In Kapitel 2 wurden diese Funktionen ausführlich beschrieben. Sie sollten allerdings nicht den Cursor in eine andere Zeile bewegen und später wieder in die Eingabezeile zurückkehren. Es könnte sonst vorkommen, daß das am Anfang der Zeile stehende Fragezeichen als ein Teil der Eingabe angesehen wird.
Wenn Daten von der Tastatur gelesen werden, wird mit jedem Tastendruck ein ATASCII-Zeichen erzeugt. Mit Hilfe der Tastatur können alle 256 Zeichenkodes erzeugt werden. In Teil D des Anhanges ist aufgeführt, welche Kodezahl bei Betätigung einer bestimmten Taste oder Tastenkombination ausgegeben wird.

Verwendung des INPUT-Befehls im Zusammenhang mit anderen Geräten

Das bisher gesagte gilt immer, ganz gleich, von welchem Gerät mit Hilfe des INPUT-Befehls Daten gelesen werden. Bei anderen Geräten hat das EOL-Zeichen dieselbe Funktion wie die RETURN-Taste bei der Tastatur. Wenn mehrere numerische Werte unter Verwen-

dung eines einzigen INPUT-Befehls gelesen werden sollen, dann dürfen diese Werte auch durch Kommas an Stelle von EOL-Zeichen voneinander getrennt werden.

Verwendung des INPUT-Befehls im Zusammenhang mit der Diskettenstation und dem Programmrecorder

Die Datenübertragung von der Diskettenstation oder dem Programmrecorder geschieht nicht zeichenweise, sondern in Blöcken. Bei der ATARI-Diskettenstation 810 enthält ein Block 125 und beim Programmrecorder enthält ein Block 128 Zeichen.
Ein Block kann sowohl einen alphanumerischen Wert beziehungsweise einen Teil davon als auch einen oder mehrere durch Kommas oder EOL-Zeichen voneinander getrennte numerische Werte enthalten. Wenn mit Hilfe eines INPUT-Befehls Daten von der Diskettenstation oder dem Programmrecorder gelesen werden sollen, dann wird dafür zunächst der im Speicher befindliche Block verwendet. Sollte der Inhalt dieses Blockes nicht ausreichen, um allen angegebenen Variablen einen Wert zuzuweisen, dann wird der nächste Block geladen.
Sobald Sie versuchen, über das Ende eines Kassetten- oder Diskettenfiles hinauszulesen, tritt ein Fehler auf.

Verwendung des INPUT-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Kanal *Kan* muß für Eingabe von der gewünschten RS-232-Schnittstelle des ATARI-Interfacemodul 850 geöffnet sein. Das funktioniert nur dann, wenn ein Programm zur Bedienung der RS-232-Schnittstellen beim Einschalten des Computers gebootet wurde. Außerdem muß die serielle Schnittstelle mit einem XIO 40-Befehl für gleichzeitige Ein- und Ausgabe eingestellt und mit Hilfe eines XIO 38-Befehls festgelegt werden, ob und wie die empfangenen Daten übersetzt werden sollen (vom ASCII- in den ATASCII-Kode). Bevor diese Aufgaben nicht erledigt sind, ist es nicht möglich, mit Hilfe des INPUT-Befehls Daten von einer seriellen Schnittstelle zu lesen.
Wenn nun von einem externen Gerät Daten zum ATARI-Interfacemodul 850 übertragen werden, dann werden diese Daten entsprechend der eingestellten Betriebsart der Schnittstelle in ATASCII-Zeichen übersetzt. Bei zwei Betriebsarten wird dabei auch das ASCII-Zeichen für "Wagenrücklauf" in das ATASCII-EOL-Zeichen übersetzt und damit das Ende einer Eingabe gekennzeichnet. In Teil D des Anhanges sind die Kodezahlen des ATASCII- und des ASCII-Kodes einander gegenübergestellt.
Danach werden die empfangenen Daten Byte für Byte an den Computer übergeben und dort entsprechend der obigen Beschreibung weiterverarbeitet.

LET = (= oder LE. =)

Mit Hilfe des LET =- oder =-Befehls wird einer Variablen ein Wert zugewiesen.

Format: [LET] *Var* = *Ausdr*
Beispiele:
```
LET A=B
LE. A$="Außenministerium"
KURS (1,N)=KURS(1,N-1)+SIN(X/Y)
ANTWORT=RICHTIG OR FALSCH
ALPHA=BETA<=GAMMA
```

Bei Ausführung dieses Befehls wird der Variablen *Var* der Wert von *Ausdr* zugewiesen. Bei

der Variablen kann es sich um eine numerische oder um eine alphanumerische Variable, um ein Feldelement oder um einen Teil der alphanumerischen Variablen handeln. *Ausdr* und *Var* müssen natürlich vom gleichen Typ sein. Es ist allerdings möglich, einer numerischen Variablen das Ergebnis eines booleschen Ausdrucks oder eines Vergleichs zuzuweisen. In dieser Variablen ist dann entweder eine 1 (wahr) oder eine 0 (falsch) gespeichert. Bei einem Vergleich darf nur auf einer Seite des Kleiner- (<), Größer- (>) oder Ungleich-Zeichens (< >) eine STR$-Funktion stehen. Das gleiche gilt auch für die CHR$-Funktion. Wenn es sich bei *Var* um einen Teil einer alphanumerischen Variablen handelt, dann wird, wenn Sie *Var* einen Wert zuweisen, auch wirklich nur dieser Teil der Variablen verändert. Der Rest der Variablen sieht nach der Ausführung des Befehls genauso aus wie vorher. Die Werte, die in einem Teil der Variablen stehen, der noch nicht benutzt wurde, sind zufallsbedingt.

LIST (L.)

Dient zur Darstellung des gerade im Speicher befindlichen Programmes beziehungsweise eines Teils davon. Mit Hilfe dieses Befehls kann aber auch das gerade im Speicher befindliche Programm oder ein Teil davon an ein bestimmtes Gerät ausgegeben werden.

Formate: LIST [*ProZeil*$_1$[,*ProZeil*$_2$]]
LIST *AusGer* [,*Prozeil*$_1$ [,*Prozeil*$_2$]]

Beispiele:
```
LIST
L. 160
L. "P:", 100,200
L. "D:TEST.BAS"
```

Bei Verwendung des ersten oben angegebenen Formates wird in der Grafikbetriebsart 0 ein Programm oder ein Teil davon mit Hilfe des Editors (Gerätename E:) auf dem Bildschirm aufgelistet. Dieses Listing kann unter Umständen etwas anders aussehen als das, was bei der Ausführung des Programmes in einer anderen Grafikbetriebsart auf dem Bildschirm dargestellt wird. Bei Verwendung des zweiten Formates wird ein Programm oder ein Teil davon an das Gerät *AusGer* übertragen. Bei diesem Gerät kann es sich um den Bildschirm (Gerätename S: oder E:), den Drucker (Gerätename P:), eine RS-232-Schnittstelle (Gerätename R[*n*]:), die Diskettenstation (Gerätename D[*n*]:*Filename*[.*Ext*]) oder den Programmrecorder (Gerätename C:) handeln.
Wenn Sie bei einem LIST-Befehl sowohl *ProZeil*$_1$ als auch *ProZeil*$_2$ angeben und beide Zeilennummern im Programm vorhanden sind, dann beginnt das Listing mit der Zeile, die die Zeilennummer *ProZeil*$_1$ trägt und es endet mit der Zeile, die die Zeilennummer *ProZeil*$_2$ trägt. Wenn keine Zeile mit der Zeilennummer *ProZeil*$_1$ im Programm vorkommt, dann beginnt das Listing mit der nächsthöheren Zeilennummer. Falls keine Zeile mit der Zeilennummer *ProZeil*$_2$ im Programm vorkommt, dann endet das Listing mit der nächstniederen Zeilennummer. Wenn Sie nur *ProZeil*$_1$ angeben, dann wird, sofern sie im Programm vorkommt, nur die Zeile aufgelistet, die diese Zeilennummer trägt. Falls Sie weder *ProZeil*$_1$ noch *ProZeil*$_2$ angeben, so wird das ganze Programm aufgelistet.

Format des Listings

Auch die Befehle, für die Sie bei der Eingabe nur Abkürzungen verwendet haben, werden bei der Ausführung eines LIST-Befehls ausgeschrieben. Falls es nicht bereits vorhanden ist, wird außerdem vor und hinter einem Variablennamen oder einem Befehlswort ein Leerzeichen eingefügt, damit das Listing besser lesbar ist. Daß eine Programmzeile nicht länger als drei Bildschirmzeilen sein darf, gilt nur bei der Eingabe, aber nicht für den LIST-Befehl. Wenn Sie bei der Eingabe einer Programmzeile alle nicht unbedingt benötigten Leer-

zeichen weglassen und für alle Befehle die entsprechenden Abkürzungen verwenden, dann kann es vorkommen, daß diese Zeile, dessen Länge bei der Eingabe durchaus innerhalb der zulässigen Grenze lag, bei der Darstellung mit Hilfe des LIST-Befehls mehr als drei Bildschirmzeilen in Anspruch nimmt. Diese Zeile kann dann allerdings nicht mehr editiert werden.
Unabhängig davon, welches Gerät angesprochen wird, erfolgt die Ausgabe eines Programmes bei Verwendung des LIST-Befehls immer im ATASCII-Kode. Ein auf diese Weise auf Kassette oder Diskette gespeichertes Programm kann mit Hilfe des ENTER-Befehls wieder gelesen werden. Es ist jedoch nicht möglich, ein unter Verwendung des LIST-Befehls gespeichertes Programm mit dem CSAVE- oder SAVE-Befehl zu laden. Einen Weg, wie Sie nicht mehr benötigte Variablen mit Hilfe des LIST- und des ENTER-Befehls aus der Variablentabelle löschen können, finden Sie in Bild 11-1.

Belegung der Ein/Ausgabekanäle

Wenn Sie ein Programm mit Hilfe des LIST-Befehls an ein bestimmtes Gerät übertragen, dann wird dafür Kanal 7 verwendet. Eine Ausnahme bildet jedoch der Bildschirm, weil hier nicht Kanal 7, sondern Kanal 0 benutzt wird. Der LIST-Befehl funktioniert auch dann, wenn Kanal 7 bereits belegt ist. Allerdings wird Kanal 7 nach der Ausführung eines LIST-Befehls geschlossen, so daß er nicht mehr für den Zweck benutzt werden kann, für den er vor der Ausführung des LIST-Befehls verwendet wurde. Außerdem schaltet der LIST-Befehl auch jegliche Tonerzeugung ab.

Verwendung des LIST-Befehls im Zusammenhang mit dem Programmrecorder

Mit Hilfe eines LIST "C:"-Befehls kann ein im Speicher befindliches Programm mit Hilfe des Programmrecorders auf Kassette abgelegt werden. Bei der Ausführung eines solchen Befehls ertönt zunächst zweimal ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die RECORD- und die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie dann etwa 20 Sekunden lang einen gleichmäßigen hohen Ton hören, auf den eine oder mehrere schnelle Tonfolgen folgen: Das Programm wird gespeichert.

Verwendung des LIST-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des LIST-Befehls ein Programm auf Diskette ablegen wollen, muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN DES DISK OPERATING SYSTEM"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf. Anderenfalls wird das im Speicher befindliche Programm zur Diskettenstation übertragen und auf Diskette abgelegt.

Verwendung des LIST-Befehls im Zusammenhang mit dem Drucker

Mit Hilfe eines LIST "P:"-Befehls können Sie ein Programm auf einem ATARI-Drucker ausdrucken. Der Drucker muß natürlich vorher eingeschaltet worden sein. Bei Verwendung des ATARI-Druckers 825 muß außerdem der Online-Schalter in der richtigen Stellung stehen und das ATARI-Interfacemodul 850, über das der Drucker angeschlossen wird, muß

ebenfalls vorher eingeschaltet worden sein. Der Drucker verfügt nicht über denselben Zeichensatz wie der Computer. Aus diesem Grund unterscheiden sich einige Zeichen eines ausgedruckten und eines auf dem Bildschirm dargestellten Listings voneinander.
Der ATARI-Drucker 825 interpretiert einige ATASCII-Zeichen als Steuerzeichen. Wenn solche Zeichen in einem Programmlisting enthalten sind (meistens in Anführungszeichen), dann führt der Drucker die entsprechende Funktion aus und ruiniert dabei höchstwahrscheinlich den Ausdruck. So etwas kann nicht passieren, wenn Sie die Steuerzeichen nicht direkt ins Programm schreiben, sondern stattdessen die CHR$-Funktion verwenden.

Verwendung des LIST-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Wenn Sie den LIST-Befehl im Zusammenhang mit einer RS-232-Schnittstelle verwenden wollen, dann muß beim Einschalten des Computers ein Programm zur Bedienung der RS-232-Schnittstellen gebootet worden sein. Falls dieses Programm nicht vorhanden ist, tritt ein Fehler auf. Außerdem muß die Schnittstelle mit Hilfe einiger XIO-Befehle initialisiert werden. Danach kann ein Programm vom Computer zu einer seriellen Schnittstelle übertragen werden. Dabei wird allerdings nicht überprüft, ob die übertragenen Daten ordnungsgemäß empfangen werden oder ob überhaupt irgendein Gerät an der benutzten Schnittstelle angeschlossen ist.

Unterbrechung des LIST-Befehls

Die Ausführung eines LIST-Befehls kann durch Drücken der SYSTEM RESET- oder der BREAK-Taste unterbrochen werden. Wenn gerade ein Programm an den Programmrecorder übertragen wurde, dann ist zwar nur ein Teil des Programmes gespeichert worden, aber dieser Teil kann unter Verwendung des ENTER-Befehls gelesen werden.
Wenn mit Hilfe des unterbrochenen LIST-Befehls ein Programm in einem Diskettenfile gespeichert werden sollte, dann ist es sehr wahrscheinlich, daß dieses File nach der Unterbrechung gar nicht existiert. Falls Sie jedoch die SYSTEM RESET- oder die BREAK-Taste drücken, wenn der Speichervorgang schon fast beendet ist, dann vervollständigt der Computer das File möglicherweise noch bevor er die Ausführung des LIST-Befehls unterbricht. Wenn Sie die SYSTEM RESET-Taste jedoch zu einem ungünstigen Zeitpunkt drücken, dann wird das Listing zerstört, das File nicht beendet und der Computer gerät in eine Endlosschleife, aus der Sie nur herauskommen, indem Sie ihn aus- und wieder einschalten.

LOAD (LO.)

Mit Hilfe des LOAD-Befehls wird ein vorher gespeichertes Programm von dem angegebenen Gerät zum Computer übertagen.

Format: LOAD *EinGer*
Beispiele: LOAD "C:"
LO. "D:PROGRAM1"
LO. PRG$

Bei Ausführung des LOAD-Befehls wird von dem angegebenen Gerät *EinGer* ein BASIC-Programm geladen. Während des Ladevorganges wird die gerade im Speicher befindliche Variablentabelle gelöscht und durch die des Programmes, das gerade geladen wird, ersetzt.
Mit Hilfe des LOAD-Befehls können nur Programme geladen werden, die unter Verwendung des SAVE-Befehls gespeichert wurden. Nicht geladen werden können Programme,

die mit Hilfe des CSAVE-Befehls gespeichert wurden, weil hier das Timing anders ist. Auch Programme, die unter Verwendung des LIST-Befehls im ATASCII-Kode abgelegt wurden, können so nicht geladen werden.
Zur Übertragung der Programme vom Programmrecorder oder der Diskettenstation zum Computer wird Kanal 7 verwendet. Der LOAD-Befehl funktioniert auch dann, wenn Kanal 7 bereits belegt ist.

LOAD und NEW

Im LOAD-Befehl ist der NEW-Befehl gewissermaßen enthalten. Bei seiner Ausführung werden alle Programmzeilen und Variablen aus dem Speicher gelöscht.
Nachdem der Ladevorgang beendet worden ist (ganz gleich, ob erfolgreich oder nicht erfolgreich beziehungsweise vollständig oder nicht vollständig), wird jegliche Tonerzeugung abgeschaltet und alle Ein-/Ausgabekanäle bis auf Kanal 0 geschlossen. Hierbei ist zu beachten, daß auch Kanal 6, der bei der Ausführung von einigen Grafikbefehlen verwendet wird, geschlossen wird.

Verwendung des LOAD-Befehls im Zusammenhang mit dem Programmrecorder

Mit Hilfe eines LOAD "C:"-Befehls wird ein auf Kassette gespeichertes Programm mit Hilfe des Programmrecorders geladen und im Speicher abgelegt. Bei der Ausführung eines solchen Befehls ertönt zunächst ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie nach einer kleinen Pause eine oder mehrere schnelle Tonfolgen hören: Das Programm wird geladen.

Verwendung des LOAD-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des LOAD-Befehls ein Programm von einer Diskette laden wollen, muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN DES DISK OPERATING SYSTEM"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf. Anderenfalls wird das angegebene Programm von der Diskette geladen und im Speicher abgelegt.

Unterbrechen des LOAD-Befehls

Die Ausführung eines LOAD-Befehls kann durch Drücken der SYSTEM RESET-Taste unterbrochen werden. Es befindet sich dann allerdings noch keine einzige Programmzeile im Speicher. Wenn Sie mit Hilfe eines LOAD-Befehls ein Programm von der Diskette laden und während der Ausführung dieses Befehls die SYSTEM RESET-Taste drücken, dann kann der Computer unter Umständen in eine Endlosschleife geraten, aus der Sie nur wieder herauskommen, indem Sie ihn aus- und wieder einschalten.
Die Ausführung eines LOAD-Befehls wird nicht unterbrochen, wenn Sie die BREAK-Taste drücken. Sie wird nur für einen kurzen Augenblick gestoppt. Dabei kann ein Teil des Programmes, das gerade geladen wird, verloren gehen.

LOCATE (LOC.)

Liest den Kode eines Zeichens oder Grafikpunktes, der sich an der angegebenen Bildschirmposition befindet.

Format: LOCATE *Spal, Zeil, NumVar*
Beispiele: LOCATE 5, 10, PUNKT
LOC. SPAL,ZEIL,KODE
LOC. PEEK(86)*256+PEEK(85),PEEK(84),Wert

Bei der Ausführung eines LOCATE-Befehls wird der Kode eines Zeichens oder Grafikpunktes gelesen, der sich an der durch *Spal* und *Zeil* gekennzeichneten Bildschirmposition befindet. Diese Kodezahl wird der Variablen *NumVar* zugewiesen.
Bei dieser Zahl handelt es sich um die Kodezahl eines Bytes. Ihr Wert liegt deshalb zwischen 0 und 255. Die Kodezahl entspricht der Zahl, die bei einem COLOR-Befehl angegeben werden müßte, um das entsprechende Zeichen beziehungsweise den entsprechenden Punkt auf dem Bildschirm darzustellen. In der Grafikbetriebsart 0 kann mit Hilfe dieser Kodezahl festgestellt werden, welches Zeichen sich an der angebenen Bildschirmposition befindet (siehe Tabelle 11-3). In den Grafikbetriebsarten 1 und 2 kann außer dem Zeichen selbst auch die Zeichenfarbe bestimmt werden (siehe Tabelle 11-4). In den Grafikbetriebsarten 3 bis 8 enthält die Kodezahl eine Inktormation darüber, welches Farbregister die Farbe des an der angegebenen Bildschirmposition befindlichen Punktes bestimmt (siehe Tabelle 11-1).

Was bei der Benutzung des LOCATE-Befehls zu beachten ist

Damit der LOCATE-Befehl ordnungsgemäß funktionieren kann, muß Kanal 6 für Eingabe von dem Bildschirm geöffnet sein. Dies geschieht bei der Ausführung eines GRAPHICS-Befehls automatisch.

Die Cursorposition

Nach der Ausführung eines LOCATE-Befehls steht der Cursor rechts von der angegebenen Bildschirmposition. Dabei wird in der Speicherzelle 84 die Zeilennummer und in den Speicherzellen 85 und 86 die Spaltennummer der neuen Cursorposition gespeichert. Der nächste Befehl, mit dessen Hilfe Daten auf den Bildschirm gebracht oder von dem Bildschirm gelesen werden, bezieht sich auf diese Cursorposition. Dies gilt jedoch nicht für den DRAWTO- und die XIO-Befehle, weil diese Befehle die in den Speicherzellen 90-92 gespeicherte Cursorposition verwenden.
Wenn sich ein LOCATE-Befehl auf die letzte Spalte einer Zeile bezieht, dann befindet sich der Cursor nach der Ausführung dieses Befehls in der ersten Spalte der nächsten Zeile. Falls sich ein LOCATE-Befehl auf die letzte Spalte der letzten Zeile des Bildschirmes bezieht, so muß der Cursor nach der Ausführung dieses Befehls neu positioniert werden. Anderenfalls wird ein Fehler auftreten, sobald das nächste Mal auf den Bildschirm zugegriffen wird.

Ausführung eines PRINT-Befehls nach einem LOCATE-Befehl

Wenn nach einem LOCATE-Befehl ein PRINT-Befehl ausgeführt wird, kann es vorkommen, daß das Zeichen oder der Punkt, dessen Kode gerade gelesen wurde, verändert wird. Um das zu verhindern, verschieben Sie den Cursor nach der Ausführung eines LOCATE-Befehls mit Hilfe des POSITIONS-Befehls um eine Stelle nach links und schreiben unter Ver-

wendung des PUT-Befehls die Kodezahl auf den Bildschirm, die Sie gerade gelesen haben.

LPRINT (LP.)

Mit Hilfe des LPRINT-Befehls werden Zeichen an den Drucker übertragen.

Format: LPRINT [*Ausdr*] [{;,} ...[*Ausdr*]]...

Beispiele:
```
LPRINT "Kundennummer";KUN
LP. R$;"summe: ";SUM1+SUM2
```

Dieser Befehl entspricht im wesentlichen dem PRINT-Befehl, nur wird hier direkt ein an den seriellen Bus angeschlossener ATARI-Drucker 820 oder 822 oder ein über die parallele Schnittstelle des ATARI-Interfacemoduls 850 angeschlossener ATARI-Drucker 825 angesprochen. Natürlich muß der Drucker und gegebenenfalls auch das ATARI-Interfacemodul 850 vorher eingeschaltet worden sein. Wenn der Drucker aus irgendwelchen Gründen nicht druckbereit ist, wird nach einer kurzen Pause eine Fehlermeldung ausgegeben.
Das Einsatzgebiet des LPRINT-Befehls ist sehr groß und entsprechend stark unterscheiden sich auch die einzelnen Befehle voneinander. LPRINT allein übergibt ein EOL-Zeichen an den Drucker. Wenn bei einem LPRINT-Befehl ein oder mehrere Ausdrücke (*Ausdr*) angegeben werden, dann werden die Werte dieser Ausdrücke ausgedruckt. Wie das Ergebnis dann aussieht, hängt davon ab, welche Art von Ausdrücken angegeben wurden und ob die Ausdrücke durch Kommas oder Semikolons voneinander getrennt sind.

Ausdrucken von Zahlen

Wenn der Wert eines numerischen Ausdruckes innerhalb gewisser Grenzen liegt, dann erfolgt der Ausdruck in der normalen Schreibweise. Wenn der Absolutwert dieses Wertes jedoch näher bei 0 liegt als 0.01 oder mehr als 10 Vorkommastellen vorhanden sind, dann erfolgt der Ausdruck in der Exponentialschreibweise. Vor negativen Werten steht ein Minuszeichen; positive Werte werden ohne Angabe eines Vorzeichens ausgedruckt.

Ausdrucken von alphanumerischen Werten

Bei einigen Druckern wird beim Empfang von bestimmten Zeichen ein anderer Zeichensatz aktiviert oder eine bestimmte Sonderfunktion ausgelöst. Näheres darüber finden Sie in Kapitel 6.

Funktion der Kommas und Semikolons

Die Ausdrücke, die bei einem LPRINT-Befehl angegeben werden können, müssen durch Kommas oder Semikolons voneinander getrennt werden. Damit kann die Größe des Zwischenraumes zwischen zwei ausgedruckten Werten beeinflußt werden. Der Wert eines auf ein Semikolon folgenden Ausdruckes wird direkt, ohne Einfügung eines Leerzeichens, hinter den zuletzt ausgedruckten Wert gedruckt. Der Wert eines auf ein Komma folgenden Ausdruckes wird hingegen erst beim nächsten Tabulator-Stop gedruckt. Zwischen diesem und dem zuletzt ausgedruckten Wert befinden sich also einige Leerzeichen.
Die Tabulator-Stops haben einen Abstand von zehn Zeichen zueinander. Sie befinden sich an der 1., 11., 21. .... Position einer jeden Zeile. Wenn jedoch ein Zeichen innerhalb von zwei Stellen vor einem Tabulator-Stop gedruckt wurde (also zum Beispiel in der 19. oder

20. Position einer Zeile), dann hat dieser Tabulator-Stop in dieser Zeile keine Funktion mehr.

Verwendung des LPRINT-Befehls im Zusammenhang mit dem ATARI-Drucker 825

Falls sie den LPRINT-Befehl im Zusammenhang mit dem ATARI-Drucker 825 verwenden, müssen Sie einiges beachten. Wenn zum Beispiel mit Hilfe eines LPRINT-Befehls mehr als 40 Zeichen gedruckt wurden, dann wird bei der Ausführung des nächsten LPRINT-Befehls immer eine neue Zeile begonnen, auch dann, wenn am Ende des ersten Befehls ein Komma oder ein Semikolon stand. Falls jedoch am Ende des ersten Befehls ein Semikolon stand und mit diesem Befehl 40 Zeichen oder weniger gedruckt wurden oder wenn am Ende des ersten Befehls ein Komma stand und 38 Zeichen oder weniger gedruckt wurden, dann wird bei der Ausführung des zweiten LPRINT-Befehls ab der 41. Position derselben Zeile weitergedruckt. Bei der Ausführung eines dritten LPRINT-Befehls wird dann eine neue Druckzeile begonnen. Wenn am Ende eines LPRINT-Befehls weder ein Komma noch ein Semikolon steht, so treten keine Besonderheiten auf.

Was bei der Benutzung des LPRINT-Befehls zu beachten ist

Zur Datenübertragung vom Computer zum Drucker wird Kanal 7 verwendet. Wenn Kanal 7 bereits belegt ist, wird eine Fehlermeldung ausgegeben. Gleichzeitig wird der Kanal geschlossen. Dadurch wird vermieden, daß beim nächsten Versuch, Daten mit Hilfe des LPRINT-Befehls an den Drucker zu übertragen, erneut ein Fehler auftritt.
Außerdem wird bei der Ausführung eines LPRINT-Befehls jegliche Tonerzeugung abgeschaltet.

NEW

Löscht das gerade im Speicher befindliche Programm und alle Variablen.

Format: NEW
Beispiel: NEW

Außerdem wird bei der Ausführung des NEW-Befehls jegliche Tonerzeugung abgeschaltet, alle Ein/Ausgabekanäle bis auf Kanal 0 geschlossen und alle folgenden Berechnungen im Bogenmaß durchgeführt (betrifft nur die trigonometrischen Funktionen).

NEXT (N.)

Beendet eine Schleife, an deren Anfang ein FOR-Befehl steht.

Format: NEXT *NumVar*
Beispiele: NEXT ZAEHL
N. J

Bei der Ausführung eines NEXT-Befehls wird *NumVar* um den beim entsprechenden FOR-Befehl angegebenen Wert erhöht. Abhängig davon, welche Parameter bei diesem Befehl sonst noch angegeben wurden, wird die Programmausführung danach entweder mit dem auf den NEXT-Befehl oder mit dem auf den entsprechenden FOR-Befehl folgenden Befehl fortgesetzt. Näheres können Sie der Beschreibung des FOR-Befehls entnehmen.

Falls *NumVar* nicht der Variablen entspricht, die bei dem zuletzt ausgeführten FOR-Befehl angegeben war, tritt ein Fehler auf.

Verwendung des NEXT-Befehls im Eingabemodus

Im Eingabemodus kann eine FOR-NEXT-Schleife nur dann ausgeführt werden, wenn der NEXT-Befehl in derselben Zeile steht wie der dazugehörige FOR-Befehl.
Wenn ein NEXT-Befehl am Anfang einer im Eingabemodus eingegebenen Zeile steht, dann untersucht das BASIC ein eventuell im Speicher befindliches Programm daraufhin, ob dort eine entsprechende FOR-NEXT-Schleife vorhanden ist, die noch nicht beendet wurde. Falls eine solche Schleife nicht gefunden wird, tritt ein Fehler auf. Ansonsten wird die Ausführung dieser Schleife fortgesetzt.

NOTE (NO.)

Mit Hilfe des NOTE-Befehls kann festgestellt werden, an welcher Stelle eines bestimmten Diskettenfiles der Zeiger dieses Files steht.

Format: NOTE #*Kan, SektVar, ByteVar*
Beispiele: NOTE #5,SEKTOR, BYTE
NO. #FILE2,S,B

Kanal *Kan* muß für ein Diskettenfile geöffnet sein. Bei der Ausführung eines NOTE-Befehls wird die Position des Zeigers dieses Diskettenfiles bestimmt. Die Nummer des Sektors, in dem sich der Zeiger gerade befindet, wird der Variablen *SektVar* und die Nummer des entsprechenden Bytes innerhalb dieses Sektors der Variablen *ByteVar* zugewiesen. Bei diesen Variablen darf es sich nicht um Feldvariablen handeln. Es spielt keine Rolle, ob das Diskettenfile zum Lesen oder Schreiben geöffnet wurde.
Unter DOS 1.0 kann der NOTE-Befehl nicht benutzt werden.

ON-GOSUB

Bei der Ausführung dieses Befehls wird, abhängig von dem Wert eines bestimmten Ausdruckes, eines der angegebenen Unterprogramme ausgeführt.

Format: ON *NumAusdr* GOSUB *ProZeil* [,*ProZeil* ...]
Beispiele: ON X GOSUB 100,200,300
ON SI GOSUB B+L*100,12000,12050,100

Wenn der Integer-Wert von *Ausdr* 1 ist, wird die Ausführung des Programmes mit der ersten angegebenen Zeilennummer (*ProZeil*) fortgesetzt; wenn der Integer-Wert 2 ist, mit der zweiten und so weiter. Beim Auftreten des nächsten darauf folgenden RETURN-Befehls springt der Computer zu dem direkt auf den ON-GOSUB-Befehl folgenden Befehl zurück.
Der Wert von *Ausdr* muß zwischen 0 und 255 liegen. Wenn er außerhalb dieser Grenzen liegt, tritt ein Fehler auf. Falls der Wert von *Ausdr* 0 oder größer als die Anzahl der angegebenen Zeilennummern ist, so wird die Programmausführung mit dem auf den ON-GOSUB-Befehl folgenden Befehl fortgesetzt.

ON-GOTO

Bei der Ausführung dieses Befehls wird, abhängig von dem Wert eines bestimmten Aus-

druckes, die Programmausführung mit einer der angegebenen Zeilennummern fortgesetzt.

Format: ON *NumAusdr* GOTO *ProZeil* [,*ProZeil* ...]
Beispiele: ON ANTWORT GOTO 1000,2000,3000,4000,5000
ON RND(0)*10 GOTO 100+WERT*10,2000,3000,3000,3000

Wenn der Integer-Wert von *Ausdr* 1 ist, wird die Ausführung des Programmes mit der ersten angegebenen Zeilennummer (*ProZeil*) fortgesetzt; wenn der Integer-Wert 2 ist, mit der zweiten und so weiter.
Der Wert von *Ausdr* muß zwischen 0 und 255 liegen. Wenn er außerhalb dieser Grenzen liegt, tritt ein Fehler auf. Falls der Wert von *Ausdr* 0 oder größer als die Anzahl der angegebenen Zeilennumern ist, so wird die Programmausführung mit dem auf den ON-GOTO-Befehl folgenden Befehl fortgesetzt.

OPEN (O.)

Weist einem bestimmten Ein/Ausgabekanal ein bestimmtes Gerät zu.

Format: OPEN #*Kan, BetrAusdr, FktAusdr, Ger*
Beispiele: OPEN #1,4,0,"C:"
O. #5,BETR,0,"D:RECHNG.DAT"
O. #2,8,0,"P:"

Bevor mit Hilfe des BASIC auf ein externes Gerät zugegriffen werden kann, muß ein Ein/Ausgabekanal (*Kan*) für dieses Gerät geöffnet werden. Wenn der angegebene Kanal (*Kan*) bereits für ein anderes Gerät geöffnet ist, tritt ein Fehler auf.
Der Wert von *BetrAusdr* bestimmt, in welcher Betriebsart das angesprochene Gerät betrieben wird (zum Beispiel Eingabe oder Ausgabe). In Tabelle 11-8 sind die Werte aufgeführt, die für *BetrAusdr* eingesetzt werden können. Der zweite Parameter des OPEN-Befehls (*FktAusdr*) wird meistens nicht benutzt. In Tabelle 11-9 ist aufgeführt, welche Werte hierfür eingesetzt werden können. In den folgenden Abschnitten wird beschrieben, wie der OPEN-Befehl im Zusammenhang mit den einzelnen Geräten funktioniert.
Mit Hilfe des letzten Parameters des OPEN-Befehls (*Ger*) wird das Gerät ausgewählt, für das der angegebene Ein/Ausgabekanal geöffnet wird. Bei *Ger* kann es sich sowohl um eine alphanumerische Konstante als auch um eine alphanumerische Variable handeln. Die möglichen Gerätenamen sind in Tabelle 11-10 aufgeführt.

Verwendung des OPEN-Befehls im Zusammenhang mit dem Programmrecorder

Der Programmrecorder kann entweder für Ein- oder für Ausgabe, aber nicht für beides gleichzeitig geöffnet sein.
Falls der Programmrecorder für Eingabe geöffnet wird, ertönt zunächst ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Der Programmrecorder braucht nun etwa 20 Sekunden, um den am Anfang eines jeden Kassettenfiles befindlichen Leader zu lesen. Noch bevor das Ende des Leaders erreicht worden ist, muß der Computer bereits zum erstenmal einen GET- oder INPUT-Befehl zum Einlesen von Daten ausgeführt haben. Wenn danach nicht weitere Befehle zum Laden von Daten folgen, wird der Programmrecorder erst einmal gestoppt.

Tabelle 11-8: Der erste Parameter des OPEN-Befehls (*BetrAusdr*)

Gerät	Wert von *BetrAusdr*	Eingestellte Betriebsart			
Programmrecorder (C:)	4	Lesen			
	8	Schreiben			
Diskettenstation (D[n]:*Filename* [.*Ext*])	4	Lesen			
	6	Lesen der Directory			
	8	Schreiben (neues File)			
	9	Schreiben (Erweitern eines Files)			
	12	Lesen und Schreiben (Verändern eines bereits bestehenden Files)			
Bildschirmeditor (E:)	8	Ausgabe auf dem Bildschirm			
	12	Eingabe von der Tastatur, Ausgabe auf dem Bildschirm			
	13	Eingabe vom Bildschirm, Ausgabe auf dem Bildschirm			
Tastatur (K:)	4	Lesen			
Drucker (P:)	8	Schreiben			
RS-232-Schnittstelle (R[*n*]:)	5	Lesen			
	8	Schreiben in Blöcken			
	9	Schreiben			
	13	Gleichzeitig Lesen und Schreiben			
		Bildschirm Löschen	Text-fenstert	Lesen	Schreiben
Bildschirm (S:)	8	Ja	Nein	Nein	Ja
	12	Ja	Nein	Ja	Ja
	24	Ja	Ja	Nein	Ja
	28	Ja	Ja	Ja	Ja
	40	Nein*	Nein	Nein	Ja
	44	Nein*	Nein	Ja	Ja
	56	Nein*	Ja	Nein	Ja
	60	Nein*	Ja	Ja	Ja

* In der Grafikbetriebsart 0 wird der Bildschirm immer gelöscht.
† In der Grafikbetriebsart 0 gibt es kein getrenntes Textfenster.

Falls der Programmrecorder für Ausgabe geöffnet wird, ertönt zunächst zweimal ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die RECORD- und die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie einen gleichmäßigen hohen Ton hören: der Leader des Files wird auf die Kassette geschrieben. Innerhalb von 30 Sekunden nach der Ausführung des OPEN-Befehls müssen entweder 128 Bytes ausgegeben oder der Ausgabekanal geschlossen werden: Anderenfalls werden keine gültigen Daten gespeichert, wodurch beim Lesen des Files ein Fehler auftreten wird.

Verwendung des OPEN-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie bei einem OPEN-Befehl als Gerätenamen ein Diskettenfile angeben, dann muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN

Tabelle 11-9: Der zweite Parameter des OPEN-Befehls (*FktAusdr*)

Gerät	Funktion	Wert von FktAusdr
Programmrecorder (C:)	Normal große Pausen zwischen den aufgezeichn. Blöcken	0
	Kurze Pausen zwischen den aufgezeichneten Blöcken	128
Diskettenstation (D[*n*]:*Filename* [.*Ext*])	Keine Wirkung	0
Bildschirmeditor (E:)	Keine Wirkung	0
Tastatur (K:)	Keine Wirkung	0
Drucker (P:)	Normale Zeichen	0
	Liegende Zeichen (ATARI 820)	83
RS-232-Schnittstelle (R[*n*]:)	Keine Wirkung	0
Bildschirm (S:)	Grafikbetriebsart 0	0
	Grafikbetriebsart 1	1
	Grafikbetriebsart 2	2
	Grafikbetriebsart 3	3
	Grafikbetriebsart 4	4
	Grafikbetriebsart 5	5
	Grafikbetriebsart 6	6
	Grafikbetriebsart 7	7
	Grafikbetriebsart 8	8

Tabelle 11-10: Der dritte Parameter des OPEN-Befehls (*Ger*)

Gerät	Gerätename
Programmrecorder	C:
Diskettenstation	D[*n*]:*Filename* [.*Ext*]
Bildschirmeditor	E:
Tastatur	K:
Drucker	P:
RS-232-Schnittstelle	R[*n*]:
Bildschirm	S:

DES DISK OPERATING SYSTEM"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf.

Je nachdem, welchen Wert Sie für *BetrAusdr* angeben, wird das angegebene File für Eingabe oder Ausgabe (hier gibt es mehrere Möglichkeiten) oder die Directory für Eingabe geöffnet (siehe Tabelle 11-8). Normalerweise können maximal drei Files gleichzeitig geöffnet sein. Falls das nicht ausreichen sollte, finden Sie in Kapitel 7 eine Möglichkeit, wie die Anzahl der Files, die gleichzeitig geöffnet sein dürfen, bis auf maximal sieben erhöht werden kann. Das funktioniert allerdings nur bei DOS 2.0S.

Auch wenn Sie einen OPEN-Befehl zum Lesen der Directory geben, wird ein Filename benötigt. Es muß aber nicht unbedingt ein File mit diesem Filenamen existieren. Das ist jedoch nicht so, wenn Sie ein bestimmtes File öffnen, um Daten aus diesem File zu lesen, das File zu erweitern oder den Inhalt des Files zu verändern. Falls hier kein File mit dem angegebenen Filenamen gefunden wird, tritt ein Fehler auf. Wenn jedoch ein OPEN-Befehl

für normale Ausgabe ausgeführt wird und ein File mit dem angegebenen Filenamen bereits existiert, dann wird dieses File gelöscht und ein neues File angelegt. Falls ein neu angelegtes File nicht ordnungsgemäß geschlossen wird, bleiben die durch dieses File belegten Sektoren unter Umständen weiterhin belegt, bis die Diskette neu formatiert wird.

Verwendung des OPEN-Befehls im Zusammenhang mit dem Drucker

Der Drucker kann nur für Ausgabe geöffnet werden. Bei der Ausführung eines entsprechenden OPEN-Befehls muß der Drucker eingeschaltet sein. Wenn sich am Drucker ein Schalter mit den beiden Schalterstellungen "ONLINE" und "LOCAL" befindet, dann muß dieser Schalter in der "ONLINE"-Stellung stehen. Falls der Drucker über das ATARI-Interfacemodul 850 angeschlossen wird, muß auch das Interfacemodul eingeschaltet sein. Sobald eine dieser Bedingungen nicht erfüllt ist, tritt bei der Ausführung des OPEN-Befehls ein Fehler auf.

Verwendung des OPEN-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Wenn Sie bei einem OPEN-Befehl als Gerätenamen eine RS-232-Schnittstelle angeben, dann muß beim Einschalten des Computers ein Programm zur Bedienung der RS-232-Schnittstellen gebootet worden sein. Falls dieses Programm nicht vorhanden ist, tritt ein Fehler auf. Es tritt jedoch kein Fehler auf, wenn an die angegebene Schnittstelle kein externes Gerät angeschlossen ist oder wenn entweder das angeschlossene Gerät oder das ATARI-Interfacemodul 850 selbst nicht eingeschaltet ist.
Außer dem OPEN-Befehl müssen eventuell noch einige XIO-Befehle zur Initialisierung der angegebenen Schnittstelle ausgeführt werden. Eine RS-232-Schnittstelle kann nicht für mehrere Ein/Ausgabekanäle zur gleichen Zeit geöffnet sein.

Verwendung des OPEN-Befehls im Zusammenhang mit dem Bildschirm

Wenn Sie als Gerätenamen S: angeben, dann wird Kanal *Kan* der Bildschirm zugewiesen. Mit Hilfe von *FktAusdr* wird die Grafikbetriebsart ausgewählt. Der Wert von *BetrAusdr* bestimmt, ob der Bildschirm gelöscht wird, ob ein Textfenster eingeblendet wird und ob der Bildschirm für Ein- oder Ausgabe oder für beides geöffnet wird (siehe Tabelle 11-8).
Jedesmal, wenn ein Ein/Ausgabekanal für den Bildschirm geöffnet wird, wird der Cursor eingeblendet, der Wert der Farbregister entsprechend ihren Voreinstellungen geändert (siehe Tabelle 11-2) und Tabulator-Stops an der 7., 15., 23., ..., 103., 111. und 119. Position einer jeden Bildschirmzeile eingefügt. In der Grafikbetriebsart 0 wird der Bildschirm grundsätzlich gelöscht. Der Cursor ist sichtbar. Sie können ihn jedoch mit Hilfe des Befehls POKE 755,0 ausblenden.

In den verschiedenen Grafikbetriebsarten wird für den Bildschirmspeicher und die Display List unterschiedlich viel Speicherplatz benötigt. Dieser Speicherplatz wird bei der Ausführung eines OPEN-Befehls bereitgestellt.

Verwendung des OPEN-Befehls im Zusammenhang mit dem Bildschirmeditor

Der Bildschirmeditor empfängt Daten von der Tastatur und gibt Daten an den Bildschirm aus (unter Verwendung der Grafikbetriebsart 0). Jedesmal, wenn ein Ein/Ausgabekanal

für den Bildschirmeditor geöffnet wird, wird die Grafikbetriebsart 0 aktiviert, der Bildschirm gelöscht, der Cursor eingeblendet, der Wert der Farbregister entsprechend ihren Voreinstellungen geändert (siehe Tabelle 11-2) und Tabulator-Stops an der 7., 15., 23., ..., 103., 111. und 119. Position einer jeden Bildschirmzeile eingefügt.
Der Wert von *BetrAusdr* bestimmt, ob der Bildschirmeditor für Ein- oder Ausgabe oder für beides geöffnet wird. Der Bildschirmeditor kann auch in einer Betriebsart betrieben werden, in der mit Hilfe von INPUT-Befehlen Daten von dem Bildschirm gelesen werden können. Der Wert, der bei Ausführung eines INPUT-Befehls in dieser Betriebsart gelesen wird, wird der logischen Zeile entnommen, in der sich der Cursor gerade befindet. Dieser Wert endet, wenn von dem Bildschirm ein EOL-Zeichen gelesen wird. Die RETURN-Taste hat in dieser Betriebsart keine Funktion.

Verwendung des OPEN-Befehls im Zusammenhang mit der Tastatur

Die Tastatur (Gerätename K:) kann nur für Eingabe geöffnet werden.

PEEK

Ist am Ende dieses Kapitels in der Beschreibung der BASIC-Funktionen aufgeführt.

PLOT (PL.)

Zeichnet an der angegebenen Position einen Punkt auf den Bildschirm.

Format: PLOT *Spal, Zeil*
Beispiele: PLOT 5,15
PL. *Spal, Zeil*

Bei der Ausführung eines PLOT-Befehls wird an der durch *Spal* und *Zeil* gekennzeichneten Bildschirmposition ein Punkt gezeichnet. Die gerade aktivierte Grafikbetriebsart bestimmt die Maximalwerte, die für *Spal* und *Zeil* eingesetzt werden können (siehe Tabelle 11-7). Der zuletzt ausgeführte COLOR-Befehl bestimmt, welches Farbregister zur Festlegung der Zeichenfarbe benutzt wird. Wenn seit dem Einschalten des Computers kein COLOR-Befehl ausgeführt wurde, wird der Punkt in derselben Farbe wie der Hintergrund gezeichnet.
Bei Verwendung eines PLOT-Befehls wird die Zeilennummer, an der ein Punkt gezeichnet wurde, in der Speicherzelle 90 und die Spaltennummer in den Speicherzellen 91 und 92 gespeichert. Ein nun folgender DRAWTO-Befehl benutzt diese Speicherzellen zur Festlegung des Anfangspunktes der Linie, die mit seiner Hilfe gezeichnet wird.

Verwendung des PLOT-Befehls in den Grafikbetriebsarten 0, 1 und 2

Genauso wie in den Grafikbetriebsarten 3 bis 8, in denen der PLOT-Befehl hautpsächlich verwendet wird, kann er auch in den Grafikbetriebsarten 0, 1 und 2 benutzt werden. An Stelle eines Punktes wird hier jedoch ein Zeichen auf dem Bildschirm dargestellt. Der zuletzt ausgeführte COLOR-Befehl bestimmt, um welches Zeichen es sich dabei handelt. In den Grafikbetriebsarten 1 und 2 wird durch diesen COLOR-Befehl außerdem die Zeichenfarbe festgelegt (siehe Tabelle 11-3 und 11-4). Wenn seit dem Einschalten des Computers noch kein COLOR-Befehl ausgeführt wurde, wird so verfahren, als wenn ein COLOR 0-Befehl gegeben worden wäre.

POINT (P.)

Setzt den Zeiger eines Diskettenfiles an eine bestimmte Position.

Format: POINT #*Kan, SektVar, ByteVar*
Beispiel: POINT #5, SEKT, BYTE
P. #FILE2,S.B

Bei der Ausführung eines POINT-Befehls wird der Zeiger eines Diskettenfiles auf das mit Hilfe der numerischen Variablen *ByteVar* angegebene Byte des durch die numerische Variable *SektVar* gekennzeichneten Sektors gesetzt. Falls dieser Sektor nicht zu dem angegebenen File gehört, tritt ein Fehler auf. Der Wert von *ByteVar* muß zwischen 0 und 125 liegen. Kanal *Kan* muß für ein Diskettenfile (Lesen, Erweitern oder Verändern; nicht Schreiben!) geöffnet sein.
Unter DOS 1.0 kann der POINT-Befehl nicht benutzt werden.

POKE (POK.)

Dient zur Speicherung eines bestimmten Wertes in einer bestimmten Speicherzelle.

Format: POKE *SpAdr, ByteAusdr*
Beispiel: POKE 756,226

Der Wert von *ByteAusdr* muß zwischen 0 und 255 liegen. Bei der Ausführung eines POKE-Befehls wird dieser Wert in der mit Hilfe von *SpAdr* angegebenen Speicherzelle abgelegt. Der POKE-Befehl hat keine Wirkung, wenn *SpAdr* nicht existiert (zum Beispiel, wenn Ihr Computer auf 16K ausgebaut ist und der Wert von *SpAdr* größer ist als 16383) oder wenn mit *SpAdr* ein Nur-Lese-Speicher (ROM) angesprochen wird.
In einigen Speicherzellen sind Informationen enthalten, die der Computer benötigt, um einwandfrei zu funktionieren. Wenn Sie den Inhalt einer solchen Speicherzelle verändern, kann es vorkommen, daß ein gerade im Speicher befindliches Programm gelöscht wird oder daß der Computer sogar in eine Endlosschleife gerät, aus der Sie nur wieder herauskommen, indem Sie ihn aus- und wieder einschalten.

POP

Löscht im Stack die Rücksprungadresse des zuletzt ausgeführten FOR-, GOSUB- oder ON-GOSUB-Befehls.

Format: POP
Beispiel: POP

Bei jeder Ausführung eines FOR-, GOSUB- oder ON-GOSUB-Befehls wird eine Rücksprungadresse im Stack gespeichert. Diese Rücksprungadresse wird bei der Ausführung eines NEXT- oder RETURN-Befehls benötigt. Mit Hilfe des POP-Befehls wird die zuletzt im Stack gespeicherte Rücksprungadresse gelöscht. Es tritt kein Fehler auf, wenn der Stack leer ist.
Im Prinzip macht der POP-Befehl aus dem zuletzt ausgeführten GOSUB- oder ON-GOSUB-Befehl einen GOTO- oder ON-GOTO-Befehl. Bei der Ausführung des nächsten auf einen POP-Befehl folgenden RETURN-Befehls erfolgt der Rücksprung nicht zu dem auf den letzten, sondern zu dem auf den vorletzten GOSUB- oder ON-GOSUB-Befehl folgenden Befehl.
Wenn ein POP-Befehl innerhalb einer FOR-NEXT-Schleife ausgeführt wird, dann wird im

weiteren Verlauf des Programmes so verfahren, als wenn der zu dieser Schleife gehörende FOR-Befehl niemals ausgeführt worden wäre.

POSITION (POS.)

Setzt den Cursor an eine bestimmte Stelle des Bildschirms.

Format: POSITION *Spal, Zeil*
Beispiele: POSITION 10,3
POS. 5,BASE+N3

Alle Befehle bis auf den DRAWTO-, PLOT- und die XIO-Befehle, mit deren Hilfe Daten auf den Bildschirm geschrieben oder vom Bildschirm gelesen werden, entnehmen die nächste Cursorposition den Speicherzellen 84 (Zeilennummer), 85 und 86 (Spaltennummer). Der Inhalt dieser Speicherzellen wird bei der Ausführung eines POSITION-Befehls verändert. Der Wert von *Spal* entspricht der neuen Spaltennummer und der Wert von *Zeil* der neuen Zeilennummer. Diese neue Cursorposition wird benutzt, sobald der Bildschirm das nächste Mal mit Hilfe eines GET-, PRINT-, PUT-, INPUT- oder LOCATE-Befehls angesprochen wird. Bei der Ausführung eines POSITION-Befehls ist auf dem Bildschirm noch keine Veränderung zu erkennen; erst wenn der Bildschirm danach mit Hilfe eines anderen Befehls angesprochen wird, bewegt sich der Cursor an die neue Position.
Es tritt kein Fehler auf, wenn Sie den Cursor mit Hilfe eines POSITION-Befehls außerhalb des Bildschirmes positionieren. Dies geschieht erst dann, wenn der Bildschirm danach unter Verwendung eines anderen Befehls angesprochen wird.

PRINT (PR. oder ?)

Dient zur Ausgabe von Daten an den Bildschirm oder ein anderes Gerät.

Format: PRINT [{ *Ausdr* / #*Kan* }] [[{ ; / , } ... [*Ausdr*]]] ...

Beispiele: PRINT "Wir schreiben...."
PR. "RESTBETRAG";RE;
? #6, "X-Achse"
? #3; A$,A,B$,B,C$,C

Das Einsatzgebiet des PRINT-Befehls ist sehr groß und entsprechend stark unterscheiden sich auch die einzelnen Befehle voneinander. Wenn der PRINT-Befehl allein steht, wird nur ein ATASCII-End-of-Line-Zeichen (EOL) ausgegeben. Wenn bei einem PRINT-Befehl einer oder mehrere Ausdrücke (*Ausdr*) angegeben werden, dann werden die Werte dieser Ausdrücke über Kanal *Kan* ausgegeben. Wie diese Werte ausgegeben werden, hängt davon ab, welche Art von Ausdrücken angegeben wurden und ob die Ausdrücke durch Kommas oder Semikolons voneinander getrennt sind. Es spielt jedoch keine Rolle, an welches Gerät die Werte ausgegeben werden.

Ausgabe von Zahlen

Wenn der Wert eines numerischen Ausdruckes innerhalb gewisser Grenzen liegt, dann erfolgt die Ausgabe in der Form, die normalerweise für Zahlen verwendet wird. Wenn der Absolutwert dieses Wertes jedoch näher bei 0 liegt als 0,01 oder mehr als zehn Vorkommastellen vorhanden sind, dann erfolgt die Ausgabe in der Exponentialform. Vor negativen Werten steht ein Minuszeichen; positive Werte werden ohne Vorzeichen ausgegeben.

Funktion des Kommas und Semikolons

Die Ausdrücke, die bei einem PRINT-Befehl angegeben werden können, müssen durch Kommas oder Semikolons voneinander getrennt werden. Damit können Sie bestimmen, wieviele Leerzeichen zwischen zwei Werten ausgegeben werden sollen. Wenn zwei Ausdrücke durch ein Semikolon voneinander getrennt sind, dann werden die Werte dieser Ausdrücke direkt, ohne Einfügung eines Leerzeichens, nacheinander ausgegeben. Wenn jedoch zwei Ausdrücke durch ein Komma voneinander getrennt sind, dann werden nach der Ausgabe des Wertes des ersten Ausdruckes soviele Leerzeichen eingefügt, bis der nächste Tabulator-Stop der logischen Zeile, die gerade ausgegeben wird, erreicht worden ist.
Genau wie bei einer Druckzeile haben auch hier die Tabulator-Stops einen Abstand von zehn Zeichen zueinander. Sie befinden sich an der 1., 11., 21. ....Position einer logischen Zeile. Wenn jedoch ein Zeichen innerhalb von zwei Stellen vor einem Tabulator-Stop ausgegeben wurde (also zum Beispiel in der 19. oder 20. Position einer logischen Zeile), dann hat dieser Tabulator-Stop in dieser Zeile keine Funktion mehr.
Mit Hilfe eines einzigen PRINT-Befehls kann sowohl eine ganze logische Zeile als auch nur ein Teil davon ausgegeben werden. Wenn hinter dem letzten der bei einem PRINT-Befehl angegebenen Ausdrücke weder ein Komma noch ein Semikolon steht, dann fügt der Computer nach der Ausgabe des Wertes dieses letzten Ausdruckes ein EOL-Zeichen ein und beendet damit die logische Zeile.
Wenn jedoch ein Komma oder ein Semikolon vorhanden ist, dann gilt dasselbe, was oben über die Trennung von zwei Ausdrücken gesagt wurde, nur daß der zweite Ausdruck hier dem ersten Ausdruck des nächsten PRINT-Befehls entspricht.

Verwendung des PRINT-Befehls im Zusammenhang mit dem Bildschirm

Wenn bei einem PRINT-Befehl kein Ein/Ausgabekanal angegeben wird oder wenn beim Öffnen von Kanal *Kan* als Gerätename S: oder E: angegeben wurde, dann wird mit Hilfe des PRINT-Befehls der Bildschirm angesprochen. Die bei der Ausführung eines PRINT-Befehls ausgegebenen Zeichen werden immer dem in der Grafikbetriebsart 0 gültigen, 256 Zeichen umfassenden Zeichensatz (siehe Tabelle 11-3) entnommen, auch dann, wenn in diesem Augenblick eine andere Grafikbetriebsart aktiviert ist. In der Grafikbetriebsart 0 werden die Zeichen so dargestellt wie sie sind. In den Grafikbetriebsarten 1 und 2 werden jedoch andere Zeichensätze verwendet und in den Grafikbetriebsarten 3 bis 8 können nur Punkte dargestellt werden. In diesen Fällen werden die mit Hilfe eines PRINT-Befehls ausgegebenen Zeichen entsprechend übersetzt.
In den Grafikbetriebsarten 1 und 2 gibt es zwei Zeichensätze. Der erste Zeichensatz enthält Großbuchstaben, Zahlen und Satzzeichen und der zweite Kleinbuchstaben und Grafikzeichen. In keinem der beiden Zeichensätze sind inverse Zeichen enthalten, aber dafür kann jedes Zeichen in vier verschiedenen Farben dargestellt werden. In Tabelle 11-4 sind beide Zeichensätze aufgeführt.
In den Grafikbetriebsarten 1 und 2 wird die Übersetzung der mit Hilfe eines PRINT-Befehls ausgegebenen Zeichen durch zwei Dinge beeinflußt. Erstens kann mit Hilfe der Speicherzelle 756 zwischen den beiden Zeichensätzen umgeschaltet werden. Bei Ausführung des Befehls POKE 756,226 wird der erste und bei Ausführung des Befehls POKE 756,224 der zweite Zeichensatz aktiviert. Zweitens bestimmt der ATASCII-Kode eines mit Hilfe des PRINT-Befehls ausgegebenen Zeichens, welches Zeichen in welcher Farbe auf dem Bildschirm dargestellt wird. Um die Zeichenübersetzung nachzuvollziehen, lokalisieren Sie zunächst in der Tabelle 11-3 das Zeichen, das mit Hilfe eines PRINT-Befehls ausgegeben wird und zwar in der Form, wie es in der Grafikbetriebsart 0 auf dem Bildschirm dargestellt werden würde. Merken Sie sich die Kodezahl dieses Zeichens. Beachten Sie dabei den Unterschied zwischen normaler und invertierter Darstellung. Suchen Sie nun dieselbe Ko-

dezahl in der Tabelle 11-4. Im Kopf der Spalte, wo Sie die Kodezahl gefunden haben, ist die Nummer des Farbregisters eingetragen, das zur Darstellung eines der beiden weiter rechts neben der gefundenen Kodezahl stehenden Zeichen benutzt wird. Welches der beiden Zeichen in der Grafikbetriebsart 1 oder 2 nun wirklich dargestellt wird, das hängt davon ab, welcher der beiden Zeichensätze aktiviert ist.
In den Grafikbetriebsarten 3 bis 8 bestimmt der ATASCII-Kode eines mit Hilfe eines PRINT-Befehls ausgegebenen Zeichens die Farbe des Punktes, der in diesen Grafikbetriebsarten an Stelle eines Zeichens dargestellt wird. In den Grafikbetriebsarten 3, 5 und 7 wird von dieser Kodezahl solange 4 subtrahiert, bis eine Zahl herauskommt, deren Wert zwischen 0 und 3 liegt. In den Grafikbetriebsarten 4, 6 und 8 wird eine gerade Kodezahl durch eine 0 und eine ungerade Kodezahl durch eine 1 ersetzt. Das Ergebnis dieser Umformungen entspricht der Zahl, die sonst bei einem COLOR-Befehl angegeben wird und genauso wie dort wird auch hier das Farbregister zur Festlegung der Zeichenfarbe ausgewählt.
Die in den Speicherzellen 84 (Zeilennummer) sowie 85 und 86 (Spaltennummer) gespeicherte Cursorposition bestimmt die Stelle, ab der die mit Hilfe eines PRINT-Befehls ausgegebenen Daten auf dem Bildschirm dargestellt werden. Die Cursorposition wird von dem DRAWTO-, GET-, INPUT-, LOCATE-, PLOT-, POSITION-, PRINT-, PUT- und den XIO-Befehlen beeinflußt.

Verwendung des PRINT-Befehls im Zusammenhang mit dem Programmrecorder

Wenn Sie mit Hilfe des PRINT-Befehls Daten an den Programmrecorder ausgeben wollen, muß vorher Kanal *Kan* für Ausgabe auf den Programmrecorder geöffnet worden sein.
Es kann vorkommen, daß mit einem PRINT-Befehl nur ein Teil eines Records ausgegeben wird. Aus diesem Grund legt der Computer zunächst einmal alle für den Programmrecorder bestimmten Daten in einem bestimmten Speicherbereich ab. Erst wenn auf diese Weise 128 Bytes gesammelt worden sind, wird der ganze Datenblock auf einmal ausgegeben. Sobald jedoch ein EOL-Zeichen in diesen Zeichenspeicher gelangt, so wird auch dann ein Block ausgegeben, wenn noch keine 128 Bytes vorhanden sind. Das 128. Byte des Blockes enthält dann eine Information darüber, wieviele Zeichen dieser Block enthält.
Wenn beim Öffnen von Kanal *Kan* festgelegt wurde, daß die Pausen zwischen den einzelnen Blöcken normal lang sein sollen, dann kann der Programmrecorder zwischen den Blöcken stoppen. Bei der Verwendung von kurzen Pausen ist das nicht möglich. Hier muß die Aufzeichnung der Daten in einem Zug erfolgen. Wenn die Daten vom Programm nicht schnell genug ausgegeben werden, treten bei der Aufzeichnung Fehler auf.

Verwendung des PRINT-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des PRINT-Befehls Daten an die Diskettenstation ausgeben wollen, muß vorher Kanal *Kan* zum Schreiben, Verändern oder Erweitern eines Diskettenfiles geöffnet worden sein.
Bei der Ausgabe von Daten an die Diskettenstation wird ähnlich wie bei der Ausgabe an den Programmrecorder vorgegangen. Auch hier geschieht die Datenübertragung blockweise. Alle für die Diskettenstation bestimmten Daten gelangen zunächst in den Speicher und werden dort zwischengespeichert bis genug Daten vorhanden sind, um einen Block zu füllen. Erst dann werden die Daten ausgegeben. Sobald jedoch ein EOL-Zeichen in diesen Zwischenspeicher gelangt, werden die Daten auch dann an die Diskettenstation übertragen, wenn noch kein ganzer Block damit gefüllt werden kann. Bei der ATARI-Diskettenstation 810 enthält jeder Block 125 Bytes.

Verwendung des PRINT-Befehls im Zusammenhang mit dem Drucker

Wenn Sie mit Hilfe des PRINT-Befehls Daten an den Drucker übertragen wollen, muß vorher Kanal *Kan* für Ausgabe an den Drucker geöffnet worden sein. Bei der Ausführung eines solchen PRINT-Befehls muß der Drucker eingeschaltet sein. Wenn sich am Drucker ein Schalter mit den beiden Schalterstellungen "ONLINE" und "LOCAL" befindet, dann muß dieser Schalter in der "ONLINE"-Stellung stehen. Falls der Drucker über das ATARI-Interfacemodul 850 angeschlossen wird, muß auch das Interfacemodul eingeschaltet sein. Sobald eine dieser Bedingungen nicht erfüllt ist, tritt ein Fehler auf.
Der Zeichensatz der meisten Drucker entspricht nicht dem in der Grafikbetriebsart 0 gültigen Zeichensatz. Keiner der ATARI-Drucker kann zum Beispiel Grafikzeichen drucken. Welches Zeichen stattdessen gedruckt wird, hängt davon ab, welcher Drucker verwendet wird. In Tabelle 6-1 und 6-2 sind die Zeichensätze der ATARI-Drucker aufgeführt.

Verwendung des PRINT-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Wenn Sie mit Hilfe eines PRINT-Befehls Daten über eine RS-232-Schnittstelle ausgeben wollen, dann muß vorher Kanal *Kan* für Ausgabe an diese Schnittstelle geöffnet worden sein. Das funktioniert nur dann, wenn ein Programm zur Bedienung der RS-232-Schnittstellen beim Einschalten des Computers gebootet wurde. Außerdem müssen eventuell noch einige XIO-Befehle zur Initialisierung der Schnittstelle ausgeführt werden. Manchmal müssen zum Beispiel die auszugebenden Daten vom ATASCII- in den ASCII-Kode übersetzt werden. Dafür wird der XIO 38-Befehl verwendet. Erst wenn all diese Aufgaben erledigt sind, können mit Hilfe eines PRINT-Befehls Daten über eine serielle Schnittstelle ausgegeben werden.
Es tritt kein Fehler auf, wenn an die angegebene Schnittstelle kein externes Gerät angeschlossen ist oder wenn entweder das angeschlossene Gerät oder das ATARI-Interfacemodul 850 selbst nicht eingeschaltet ist.
Wenn nun mit Hilfe eines PRINT-Befehls Daten zum ATARI-Interfacemodul 850 übertragen werden, dann werden diese Daten entsprechend der eingestellten Betriebsart der Schnittstelle in ASCII-Zeichen übersetzt. Bei zwei Betriebsarten wird dabei auch das ATASCII-EOL-Zeichen in das ASCII-Zeichen für "Wagenrücklauf" übersetzt. In Teil D des Anhanges sind die Kodezahlen des ATASCII- und des ASCII-Kodes einander gegenübergestellt.

PUT (PU.)

Dient zur Ausgabe einer Zahl über den angegebenen Ein/Ausgabekanal.

Format: PUT #*Kan, NumAusdr*
Beispiele: PUT #1,NR
PU. #KA,X

Kanal *Kan* muß für Ausgabe geöffnet sein. Bei der Ausführung eines PUT-Befehls wird der Wert von *NumAusdr* über diesen Kanal ausgegeben. Der Wert wird gerundet, wenn es sich dabei um eine nichtganzzahlige Zahl handelt. Außerdem muß der Wert von *NumAusdr* zwischen 0 und 255 liegen. Wenn das nicht der Fall ist, wird solange 256 subtrahiert, bis das Ergebnis zwischen 0 und 255 liegt.

Verwendung des PUT-Befehls im Zusammenhang mit dem Programmrecorder

Wenn Sie mit Hilfe des PUT-Befehls Daten an den Programmrecorder ausgeben wollen, muß vorher Kanal *Kan* für Ausgabe auf den Programmrecorder geöffnet worden sein. Mit einem PUT-Befehl kann nur ein Teil eines Records ausgegeben werden. Aus diesem Grund legt der Computer zunächst einmal alle für den Programmrecorder bestimmten Daten in einem bestimmten Speicherbereich ab. Erst wenn auf diese Weise 128 Bytes gesammelt worden sind, wird der ganze Datenblock auf einmal ausgegeben. Sobald jedoch ein EOL-Zeichen in diesen Zwischenspeicher gelangt, so wird auch dann ein Block ausgegeben, wenn noch keine 128 Bytes vorhanden sind. Das 128. Byte des Blockes enthält dann eine Information darüber, wieviele Zeichen dieser Block enthält.
Wenn beim Öffnen von Kanal *Kan* festgelegt wurde, daß die Pausen zwischen den einzelnen Blöcken normal lang sein sollen, dann kann der Programmrecorder zwischen den Blöcken stoppen. Bei der Verwendung von kurzen Pausen ist das nicht möglich. Hier muß die Aufzeichnung der Daten in einem Zug erfolgen. Wenn die Daten vom Programm nicht schnell genug ausgegeben werden, treten bei der Aufzeichnung Fehler auf.

Verwendung des PUT-Befehls im Zusammenhang mit der Diskettenstation

Mit Hilfe des PUT-Befehls können auch Daten in einem für Ausgabe geöffneten Diskettenfile gespeichert werden. Bei der Ausführung eines PUT-Befehls wird ein einzelnes Byte ausgegeben. Die unter Verwendung eines PUT-Befehls auf Diskette gespeicherten Daten können mit Hilfe eines GET-Befehls wieder gelesen werden. Der INPUT-Befehl kann hier nicht verwendet werden.
Die Datenübertragung vom Computer zur Diskettenstation geschieht nicht zeichenweise, sondern blockweise. Jeder Block enthält die Daten eines Sektors der Diskette. Die mit Hilfe des PUT-Befehls ausgegebenen Zeichen gelangen zunächst in den Diskettenbuffer. Wenn der Buffer voll ist, wird der ganze Datenblock auf einmal zur Diskettenstation übertragen. Bei der Ausführung eines CLOSE-Befehls wird auch der Inhalt eines nur teilweise gefüllten Diskettenbuffers auf die Diskette geschrieben. Bei der ATARI-Diskettenstation 810 enthält jeder Block 125 Bytes.

Verwendung des PUT-Befehls im Zusammenhang mit dem Bildschirm

Wenn Kanal *Kan* für Ausgabe auf den Bildschirm (Gerätename S: oder E:) geöffnet ist, dann wird bei der Ausführung eines PUT-Befehls an der momentanen Cursorposition ein Punkt oder ein Zeichen dargestellt. In der Grafikbetriebsart 0 dient der Wert von *NumAusdr* zur Festlegung des Zeichens, das an der momentanen Cursorposition dargestellt werden soll (siehe Tabelle 11-3), in den Grafikbetriebsarten 1 und 2 zur Festlegung des Zeichens und der Zeichenfarbe (siehe Tabelle 11-4) und in den Grafikbetriebsarten 3 bis 8 zur Festlegung des Farbregisters, das die Zeichenfarbe des Punktes, der in diesen Grafikbetriebsarten an der momentanen Cursorposition dargestellt wird, bestimmt (siehe Tabelle 11-1).
Nach der Benutzung des PUT-Befehls im Zusammenhang mit dem Bildschirm wird die Position des Cursors um eine Stelle nach rechts verschoben. Dabei wird in der Speicherzelle 84 die Zeilennummer und in den Speicherzellen 85 und 86 die Spaltennummer der neuen Cursorposition gespeichert. Der nächste Befehl, mit dessen Hilfe Daten auf den Bildschrim gebracht oder von dem Bildschirm gelesen werden, bezieht sich auf diese Cursorposition. Dies gilt jedoch nicht für den DRAWTO- und die XIO-Befehle, weil diese Befehle die in den Speicherzellen 90-92 gespeicherte Cursorposition verwenden.
Wenn sich der Cursor in der letzten Spalte einer Zeile befindet, wird er nach der Ausführung eines PUT-Befehls in die erste Spalte der nächsten Zeile verschoben. Falls sich der

Cursor in der letzen Spalte der letzten Zeile befand, muß er nach der Ausführung eines PUT-Befehls neu positioniert werden. Anderenfalls wird ein Fehler auftreten, sobald das nächste Mal auf den Bildschirm zugegriffen wird.

Verwendung des PUT-Befehls im Zusammenhang mit dem Drucker

Wenn Sie mit Hilfe des PUT-Befehls Daten an den Drucker übertragen wollen, muß vorher Kanal *Kan* für Ausgabe an den Drucker geöffnet worden sein. Bei der Ausführung eines solchen PUT-Befehls muß der Drucker eingeschaltet sein. Wenn sich am Drucker ein Schalter mit den beiden Schalterstellungen "ONLINE" und "LOCAL" befindet, dann muß dieser Schalter in der "ONLINE"-Stellung stehen. Falls der Drucker über das ATARI-Interfacemodul 850 angeschlossen wird, muß auch das Interfacemodul eingeschaltet sein. Sobald eine dieser Bedingungen nicht erfüllt ist, tritt ein Fehler auf.

Verwendung des PUT-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Wenn Sie mit Hilfe eines PUT-Befehls Daten über eine RS-232-Schnittstelle ausgeben wollen, dann muß vorher Kanal *Kan* für Ausgabe an diese Schnittstelle geöffnet worden sein. Das funktioniert nur dann, wenn ein Programm zur Bedienung der RS-232-Schnittstellen beim Einschalten des Computers gebootet wurde. Außerdem müssen eventuell noch einige XIO-Befehle zur Initialisierung der Schnittstelle ausgeführt werden. Manchmal müssen zum Beispiel die auszugebenden Daten vom ATASCII- in den ASCII-Kode übersetzt werden. Dafür wird der XIO 38-Befehl verwendet. Erst wenn all diese Aufgaben erledigt sind, können mit Hilfe eines PUT-Befehls Daten über die angegebene Schnittstelle ausgegeben werden.
Es tritt kein Fehler auf, wenn an die angegebene Schnittstelle kein externes Gerät angeschlossen ist oder wenn entweder das angeschlossene Gerät oder das ATARI-Interfacemodul 850 selbst nicht eingeschaltet ist.

RAD

Schaltet von Grad auf Bogenmaß um.

Format: RAD
Beispiel: RAD

Nach der Ausführung des RAD-Befehls betrachtet das BASIC die Argumente aller nun folgenden trigonometrischen Funktionen als Bogenmaßangaben. Mit Hilfe des DEG-Befehls kann wieder auf Grad umgeschaltet werden. Nach dem Einschalten des Computers, nach Drücken der SYSTEM RESET-Taste oder nach der Ausführung eines NEW- oder RUN-Befehls wird immer im Bogenmaß gerechnet.

READ (REA.)

Weist die bei einem DATA-Befehl angegebenen Werte den angegebenen Variablen zu.

Format: READ *Var* [,var ...]
Beispiele: REA.NAME$, ABT$, PERSNR
REA. ALPHA, BETA, WINK$

Um festzulegen, welcher der bei einem DATA-Befehl angegebenen Werte welcher Variablen eines READ-Befehls zugewiesen wird, wird ein Zeiger benutzt. Nach einem Programmstart oder nach der Ausführung eines RESTORE-Befehls steht dieser Zeiger auf dem ersten Wert des ersten DATA-Befehls des gerade im Speicher befindlichen Programmes. Jedesmal, nachdem ein Wert gelesen worden ist, rückt dieser Zeiger zum nächsten Wert vor.
Natürlich muß der Typ der Variablen dem Typ der angegebenen Werte entsprechen. Eine Zahl kann sowohl einer numerischen als auch einer alphanumerischen Variablen zugewiesen werden, aber bei dem Versuch, einer numerischen Variablen einen alphanumerischen Wert zuzuweisen, wird ein Fehler auftreten.
Ein DATA-Befehl muß nicht ausgeführt werden, um auf ihn mit Hilfe eines READ-Befehls zugreifen zu können. Wenn versucht wird, mehr Daten zu lesen, als überhaupt vorhanden sind, tritt ein Fehler auf.

Verwendung des READ-Befehls im Eingabemodus

Im Eingabemodus können solange READ-Befehle ausgeführt werden, bis alle DATA-Befehle des gerade im Speicher befindlichen Programmes gelesen worden sind. Danach tritt ein Fehler auf.

REM (R. oder .)

Mit Hilfe des REM-Befehls können Erklärungen oder Kommentare in ein Programm eingefügt werden.

Format: REM *Kommentar*
Beispiele: REM Unterprogramm/Eingabe
R. Ergebnis berechnen
. Auf Eingabe (J oder N) warten

Bei *Kommentar* handelt es sich um eine beliebige Anzahl von Zeichen. Es dürfen jedoch nicht mehr Zeichen verwendet werden als in einer Programmzeile Platz finden.
REM-Befehle erscheinen zwar in Programmlistings, aber bei der Ausführung eines REM-Befehls geschieht absolut gar nichts. Ein REM-Befehl kann entweder in einer Zeile allein oder als letzter Befehl in einer Zeile stehen.

Wenn ein REM-Befehl nicht als letzter Befehl in einer Zeile steht, dann werden alle darauf folgenden Befehle als Teil des REM-Befehls angesehen und aus diesem Grund niemals ausgeführt.

RESTORE (RES.)

Dient zum Rücksetzen des für die DATA-Befehle zuständigen Zeigers.

Format: RESTORE [*ProZeil*]
Beispiele: RESTORE
RES. 140

Mit Hilfe des RESTORE-Befehls kann festgelegt werden, welcher Wert bei der Ausführung des nächsten READ-Befehls als erstes gelesen wird. Wenn keine Zeilennummer (*ProZeil*) angegeben wird, steht der Zeiger nach der Ausführung des RESTORE-Befehls auf dem ersten Wert des ersten DATA-Befehls des Programmes. Falls eine Zeilennummer vorhanden

ist, steht der Zeiger auf dem ersten Wert des in dieser Zeile befindlichen oder des ersten auf diese Zeile folgenden DATA-Befehls.

RETURN (RET.)

Bei der Ausführung eines RETURN-Befehls erfolgt ein Rücksprung zu dem unmittelbar auf den zuletzt ausgeführten GOSUB- oder ON-GOSUB-Befehl folgenden Befehl.

Format: RETURN
Beispiele: RETURN
RET.

Die Rücksprungadresse wird dem Stack entnommen. Wenn vor einem RETURN-Befehl ein POP-Befehl ausgeführt wird, dann wird die Programmausführung nicht mit dem auf den letzten, sondern mit dem auf den vorletzten GOSUB- oder ON-GOSUB-Befehl folgenden Befehl fortgesetzt.
Wenn in einem Programm mehr RETURN- und POP- als GOSUB- und ON-GOSUB-Befehle ausgeführt werden, tritt ein Fehler auf.

RUN (RU.)

Der RUN-Befehl dient zum Starten des im Speicher befindlichen Programmes. Es kann jedoch auch ein Programm von einem bestimmten Gerät geladen und dann gestartet werden.

Format: RUN [*EinGer*]
Beispiele: RUN
RUN "C:"
RU. P$
RU. "D2:BUDGET.BAS"

Die Programmausführung beginnt mit der Zeile, die die niedrigste Zeilennummer trägt. Bei der Ausführung eines RUN-Befehls wird jegliche Tonerzeugung abgeschaltet, alle Ein/Ausgabekanäle geschlossen und auf Bogenmaß umgeschaltet.

Laden eines Programmes mit Hilfe des RUN-Befehls

Wenn bei einem RUN-Befehl ein Gerätename (*EinGer*) angegeben wird, dann wird von diesem Gerät ein BASIC-Programm geladen und automatisch gestartet.
Mit Hilfe des RUN-Befehls können nur Programme geladen werden, die unter Verwendung des SAVE-Befehls gespeichert wurden. Nicht geladen werden können Programme, die mit Hilfe des CSAVE-Befehls gespeichert wurden, weil hier das Timing anders ist. Auch Programme, die unter Verwendung des LIST-Befehls im ATASCII-Kode abgelegt wurden, können so nicht geladen werden.
Während des Ladevorganges wird der gesamte Programmspeicher des Computers gelöscht und die gerade im Speicher befindliche Variablentabelle durch die des Programmes, das gerade geladen wird, ersetzt.
Zur Übertragung der Programme vom Programmrecorder oder der Diskettenstation zum Computer wird Kanal 7 verwendet. Der RUN-Befehl funktioniert auch dann, wenn Kanal 7 bereits belegt ist. Nach Beendigung des Ladevorganges wird Kanal 7 allerdings geschlossen.

Verwendung des RUN-Befehls im Zusammenhang mit dem Programmrecorder

Mit Hilfe eines RUN "C:"-Befehls wird ein auf Kassette gespeichertes Programm mit Hilfe des Programmrecorders geladen und im Speicher abgelegt. Bei der Ausführung eines solchen Befehls ertönt zunächst ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie nach einer kleinen Pause eine oder mehrere schnelle Tonfolgen hören: Das Programm wird geladen.

Verwendung des RUN-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des RUN-Befehls ein Programm von einer Diskette laden und ausführen wollen, dann muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN DES DISK OPERATING SYSTEM"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf. Anderenfalls wird das angegebene Programm von der Diskette geladen, im Speicher abgelegt und automatisch gestartet.

Unterbrechen des RUN-Befehls

Die Ausführung eines bereits gestarteten Programmes kann durch Drücken der SYSTEM RESET-Taste unterbrochen werden. Auch der Ladevorgang kann durch Drücken der SYSTEM RESET-Taste vorzeitig beendet werden. Es befindet sich dann allerdings noch keine einzige Programmzeile im Speicher. Wenn Sie mit Hilfe eines RUN-Befehls ein Programm von der Diskette laden und während dieses Ladevorganges die SYSTEM RESET-Taste drücken, dann kann der Computer unter Umständen in eine Endlosschleife geraten, aus der Sie nur wieder herauskommen, indem Sie ihn aus- und wieder einschalten.
Auch durch Drücken der BREAK-Taste kann die Ausführung eines Programmes unterbrochen werden. Es ist jedoch nicht möglich, den Ladevorgang durch Drücken der BREAK-Taste vorzeitig zu beenden. Er wird nur für einen kurzen Augenblick gestoppt. Dabei kann ein Teil des Programmes, das gerade geladen wird, verloren gehen.

SAVE (S.)

Dient zur Übertragung eines im Speicher befindlichen BASIC-Programmes an ein bestimmtes Gerät.

Format: SAVE *AusGer*
Beispiele: SAVE "C:"
S. "D:PROG1"
S. PROG$

Bei der Ausführung eines SAVE-Befehls wird das im Speicher befindliche BASIC-Programm an das angegebene Gerät *AusGer* übertragen. Bei diesem Gerät handelt es sich meistens um den Programmrecorder (Gerätename C:) oder die Diskettenstation (Gerätename D[*n*]:*Filename*[.*Ext*]). Da die mit Hilfe des SAVE-Befehls ausgegebenen Daten in einer ganz besonderen Art kodiert sind, können Sie von anderen Geräten, wie zum Beispiel dem Drucker oder dem Bildschirm nicht sinnvoll verarbeitet werden. Je nachdem, welches Gerät Sie an eine RS-232-Schnittstelle (Gerätename R[*n*]:) angeschlossen ha-

ben, kann es jedoch unter Umständen sinnvoll sein, den SAVE-Befehl im Zusammenhang mit dieser Schnittstelle zu verwenden.
Ein mit Hilfe des SAVE-Befehls gespeichertes Programm kann nur unter Verwendung des LOAD- oder des RUN-Befehls wieder geladen werden. Der CLOAD- oder der ENTER-Befehl kann hier nicht verwendet werden. Bei der Speicherung eines Programmes mit Hilfe des SAVE-Befehls wird auch die gerade im Speicher befindliche Variablentabelle mitgespeichert.

Belegung der Ein/Ausgabekanäle

Zur Übertragung eines mit Hilfe des SAVE-Befehls ausgegegebenen Programmes vom Computer zu dem angegebenen Gerät wird Kanal 7 verwendet. Der SAVE-Befehl funktioniert auch dann, wenn Kanal 7 bereits geöffnet ist. Nach Beendigung der Datenübertragung wird Kanal 7 allerdings geschlossen. Außerdem wird bei der Ausführung eines SAVE-Befehls jegliche Tonerzeugung abgeschaltet.

Verwendung des SAVE-Befehls im Zusammenhang mit dem Programmrecorder

Mit Hilfe eines SAVE "C:"-Befehls kann ein im Speicher befindliches Programm mit Hilfe des Programmrecorders auf Kassette abgelegt werden. Bei der Ausführung eines solchen Befehls ertönt zunächst zweimal ein Kontrollton. Das ist für Sie das Zeichen, eine Kassette in den Programmrecorder einzulegen und mit Hilfe der ADVANCE- (schneller Vorlauf) und der REWIND-Taste (schneller Rücklauf) an der richtigen Stelle zu positionieren. Drücken Sie danach die RECORD- und die PLAY-Taste und zum Schluß irgendeine Taste der Tastatur (außer BREAK). Wenn der Lautstärkeregler Ihres Fernsehgerätes aufgedreht ist, werden Sie dann etwa 20 Sekunden lang einen gleichmäßigen hohen Ton hören, auf den eine oder mehrere schnelle Tonfolgen folgen: Das Programm wird gespeichert.

Verwendung des SAVE-Befehls im Zusammenhang mit der Diskettenstation

Wenn Sie mit Hilfe des SAVE-Befehls ein Programm auf Diskette ablegen wollen, muß vorher das Disk Operating System gebootet worden sein (siehe auch Kapitel 2 "BOOTEN DES DISK OPERATING SYSTEMS"). Falls sich kein Disk Operating System im Speicher befindet, tritt ein Fehler auf. Anderenfalls wird das im Speicher befindliche Programm zur Diskettenstation übertragen und auf Diskette abgelegt. Falls sich auf der in der angegebenen Diskettenstation eingelegten Diskette bereits ein File mit dem angegebenen Namen befindet, so wird zunächst dieses File gelöscht und erst dann das Programm gespeichert.

Verwendung des SAVE-Befehls im Zusammenhang mit einer RS-232-Schnittstelle

Wenn Sie den SAVE-Befehl im Zusammenhang mit einer RS-232-Schnittstelle verwenden wollen, dann muß beim Einschalten des Computers ein Programm zur Bedienung der RS-232-Schnittstellen gebootet worden sein. Falls dieses Programm nicht vorhanden ist, tritt ein Fehler auf. Außerdem muß die Schnittstelle mit Hilfe einiger XIO-Befehle initialisiert werden. Danach kann ein Programm vom Computer zu einer seriellen Schnittstelle übertragen werden. Dabei wird allerdings nicht überprüft, ob die übertragenen Daten ordnungsgemäß empfangen werden oder ob überhaupt irgendein Gerät an der benutzten Schnittstelle angeschlossen ist.

Unterbrechen des SAVE-Befehls

Die Ausführung eines SAVE-Befehls kann jederzeit durch Drücken der BREAK- oder der SYSTEM RESET-Taste unterbrochen werden. Das Programm ist dann allerdings nur unvollständig gespeichert und kann deshalb nicht mit Hilfe eines LOAD- oder RUN-Befehls geladen werden.

SETCOLOR (SE.)

Bestimmt die Farbe und die Helligkeit, die bei der Benutzung eines bestimmten Farbregisters dargestellt wird.

Format: SETCOLOR *RegAusdr, FarbAusdr, HellAusdr*
Beispiele: SETCOLOR 2, 2, 2
SE. NR, FARB, HELL

Der Wert von *RegAusdr* muß zwischen 0 und 4 liegen. Mit seiner Hilfe wird eines von fünf Farbregistern ausgewählt.
Der Wert von *FarbAusdr* muß zwischen 0 und 15 liegen. Mit seiner Hilfe wird eine der 16 in Tabelle 11-11 aufgeführten Farben ausgewählt.
HellAusdr dient zur Festlegung der Helligkeit der ausgewählten Farbe. Der Wert von *HellAusdr* muß gerade sein und zwischen 0 und 14 liegen. Bei Verwendung einer ungeraden Zahl ist die Helligkeit dieselbe wie bei der nächstniedrigeren geraden Zahl. Wenn zum Beispiel für *HellAusdr 3* eingesetzt wird, dann sieht das Ergebnis genauso aus, als wenn 2 eingesetzt worden wäre.
Die Werte dieser drei Ausdrücke werden gerundet, wenn es sich dabei um nichtganzzahlige Zahlen handelt. Der Wert von *FarbAusdr* und *HellAusdr* darf zwischen 0 und 65535 liegen. Wenn er jedoch größer als 15 ist, dann wird solange 16 subtrahiert, bis das Ergebnis zwischen 0 und 15 liegt.

Tabelle 11-11: Die beim Einsetzen einer bestimmten Zahl für *FarbAusdr* erzeugten Farben (SETCOLOR-Befehl)

Wert von *FarbAusdr*	Farbe*
0	schwarz bis weiß
1	braun bis goldfarben
2	orange bis gelb
3	terrakotta bis rosa
4	dunkelrot bis magenta
5	violett bis hellblau
6	indigo bis weiß
7	himmelblau
8	königsblau bis babyblau
9	türkis
10	ultramarin bis hellblau
11	dunkelblau bis aquamarin
12	meergrün bis türkis
13	waldgrün bis hellgrün
14	oliv
15	khaki bis gelb

* Welche Farbe in Wirklichkeit dargestellt wird, hängt sowohl von der Einstellung des Fernsehgerätes als auch davon ab, welche Helligkeitsstufe gewählt wurde.

Nach dem Drücken der SYSTEM RESET-Taste oder der Ausführung eines DOS- oder GRAPHICS-Befehls werden die Werte der Farbregister entsprechend ihren Voreinstellungen geändert (siehe Tabelle 11-2).

Die Farbregister

In ATARI-BASIC gibt es fünf verschiedene Speicherzellen, mit deren Hilfe Farben auf dem Bildschirm dargestellt werden können. Diese Speicherzellen nennt man Farbregister.
In den meisten Grafikbetriebsarten wird mit Hilfe eines COLOR-Befehls bestimmt, welches Farbregister zur Festlegung der Farbe von Zeichen, Punkten und Linien benutzt wird. In Verbindung mit dem SETCOLOR-Befehl kann damit die Farbe, in der auf den Bildschirm gezeichnet wird, festgelegt werden. In Tabelle 11-12 ist aufgeführt, welches Farbregister in den verschiedenen Grafikbetriebsarten welche Funktion erfüllt.
Es gibt fünf Farbregister, die mit den Zahlen von 0 bis 4 gekennzeichnet sind. Tabelle 11-1 können Sie entnehmen, welcher COLOR-Befehl in einer bestimmten Grafikbetriebsart ausgeführt werden muß, damit ein bestimmtes Farbregister angesprochen wird. Daraus ergibt sich zum Beispiel, daß mit Hilfe des Befehls COLOR 2 in den Grafikbetriebsarten 3, 5 und 7 das Farbregister 1 ausgewählt wird.

Tabelle 11-12: Funktion der Farbregister in den verschiedenen Grafikbetriebsarten

Grafikbetriebsart	Farbregister				
	0	1	2	3	4
0	–	Z*	Z, Hi	–	Ra
1	Z	Z	Z	Z	Hi, Ra
2	Z	Z	Z	Z	Hi, Ra
3	P	P	P	–	P, Hi, Ra
4	P	–	–	–	P, Hi, Ra
5	P	P	P	–	P, Hi, Ra
6	P	–	–	–	P, Hi, Ra
7	P	P	P	–	P, Hi, Ra
8	–	P*	P, Hi	–	Ra

Anmerkung: Z = Zeichen, P = Punkt oder Linie, Hi = Hintergrund, Ra = Umrandung, * Bestimmt nur die Helligkeit; die Farbe entspricht der des Hintergrundes

SOUND (SO.)

Schaltet eine Stimme ein oder aus und bestimmt gleichzeitig die Tonhöhe, die Tonreinheit und die Lautstärke dieser Stimme.

Format: SOUND *StimmAusdr, HöhAusdr, ReinAusdr, LautAusdr*
Beispiele: SO. ST1, H, R, L2

Mit Hilfe eines Sound-Befehls wird festgelegt, welchen Ton eine der vier Stimmen des ATARI-Computers erzeugen soll. Die Stimme wird durch *StimmAusdr* ausgewählt. Der Wert von *StimmAusdr* muß zwischen 0 und 3 liegen.
Zur Festlegung der Tonhöhe dient *HöhAusdr*. Der Wert von *HöhAusdr* darf zwischen 0 (höchster Ton) und 255 (tiefster Ton) liegen.
ReinAusdr bestimmt die Tonreinheit. Der Wert von *ReinAusdr* muß gerade sein und zwischen 0 und 14 liegen. Reine Töne werden erzeugt, wenn hier eine 10 oder eine 14 einge-

setzt wird. Wenn der Wert von *HöhAusdr* zwischen 126 und 255 liegt, dann spielt es keine Rolle, ob für *ReinAusdr* eine 0 oder eine 4 eingesetzt wird, weil die damit erzeugten Töne fast gleich sind. Falls der Wert von *ReinAusdr* ungerade und die angegebene Stimme bereits eingeschaltet ist, dann wird sie bei der Ausführung dieses SOUND-Befehls ausgeschaltet.
Mit Hilfe von *LautAusdr* wird die Lautstärke der angesprochenen Stimme festgelegt. Der Wert von *LautAusdr* muß zwischen 0 (keine Tonerzeugung) und 15 (größte Lautstärke) liegen. Es tritt zwar kein Fehler auf, wenn der Wert von *LautAusdr* größer als 15 ist (maximal 65535), aber es wird dann kein Ton erzeugt. Wenn der Wert von *HöhAusdr, ReinAusdr* und *LautAusdr* 0 ist, wird die angesprochene Stimme abgeschaltet. Falls nun ein SOUND-Befehl folgt, bei dem der Wert von *ReinAusdr* ungeradzahlig ist, dann wird ein kurzer Impuls erzeugt. Wenn zwei solcher SOUND-Befehle abwechselnd ausgeführt werden, dann ist eine Serie von Impulsen zu hören.
Falls der Wert von einem der im Zusammenhang mit dem Sound-Befehl verwendeten Ausdrücke nicht ganzzahlig ist, dann wird dieser Wert gerundet. Es tritt kein Fehler auf, wenn der Wert von *HöhAusdr* oder *ReinAusdr* größer als 15 ist (maximal 65535). Von diesem Wert wird dann solange 16 subtrahiert, bis das Ergebnis zwischen 0 und 15 liegt.
Es werden alle vier Stimmen abgeschaltet, wenn Sie die SYSTEM RESET-Taste drücken oder wenn einer der folgenden Befehle ausgeführt wird: CLOAD, CSAVE, DOS, END, ENTER, LIST (außer, wenn im Zusammenhang mit dem Bildschirm verwendet), LOAD, NEW, RUN, SAVE.

Tabelle 11-13: Erzeugung einer bestimmten Note durch Angabe einer bestimmten Zahl für *HöhAusdr*

Wert von *HöhAusdr*	Note	Wert von *HöhAusdr*	Note
29	C	91	F
31	B	96	E
33	$A^{\#}/B^{b}$	102	$D^{\#}/E^{b}$
35	A	108	D
37	$G^{\#}/A^{b}$	114	$C^{\#}/D^{b}$
40	G	121	C
42	$F^{\#}/G^{b}$	128	B
45	F	136	$A^{\#}/B^{b}$
47	E	144	A
50	$D^{\#}/E^{b}$	153	$G^{\#}/A^{b}$
53	D	162	G
57	$C^{\#}/D^{b}$	173	$F^{\#}/B^{b}$
60	C	182	F
64	B	193	E
68	$A^{\#}/B^{b}$	204	$D^{\#}/E^{b}$
72	A	217	D
76	$G^{\#}/A^{b}$	230	$C^{\#}/D^{b}$
81	G	243	C
85	$F^{\#}/G^{b}$		

STATUS (ST.)

Mit Hilfe des STATUS-Befehls kann festgestellt werden, mit welchem Ergebnis die zuletzt im Zusammenhang mit dem angegebenen Kanal ausgeführte Ein- oder Ausgabeoperation abgeschlossen wurde.

Format: STATUS #*Kan, NumVar*
Beispiele: STATUS #5, STAT
ST. #C, S

Bei der Ausführung eines STATUS-Befehls wird der Variablen *NumVar* der Statuskode der letzten im Zusammenhang mit dem Kanal *Kan* ausgeführten Ein- oder Ausgabeoperation zugewiesen. Wenn der Wert dieser Kodezahl größer als 128 ist, dann ist dabei ein Fehler aufgetreten. Die Statuskodes sind in Teil B des Anhanges aufgeführt.
Wenn es sich bei *NumVar* um ein Feldelement handelt, tritt bei der Ausführung des STATUS-Befehls zwar kein Fehler auf, jedoch wird das der Fall sein, wenn später noch einmal auf dieses Feld zugegriffen wird. Nach der Ausführung eines STATUS-Befehls muß das Feld nämlich neu dimensioniert werden.

STOP (STO.)

Dient zur Unterbrechung eines BASIC-Programmes.

Format: STOP
Beispiele: STOP
STO.

Bei der Ausführung eines STOP-Befehls unterbricht der Computer das laufende Programm und aktiviert den Eingabemodus (Grafikbetriebsart 0). Auf dem Bildschirm wird die Meldung STOPPED AT LINE *Zeilnr* angezeigt, wobei für *Zeilnr* die Nummer der Zeile eingesetzt wird, in der sich der STOP-Befehl befindet. Wenn der STOP-Befehl im Eingabemodus ausgeführt wurde, erscheint keine Zeilennummer.
Mit Hilfe eines CONT-Befehls kann die Programmausführung mit der Zeile fortgesetzt werden, die direkt auf die Zeile folgt, in der die Unterbrechung auftrat. Die Ausführung einer im Eingabemodus eingegebenen Zeile kann nicht fortgesetzt werden.
Bei der Ausführung eines STOP-Befehls wird weder die Tonerzeugung abgeschaltet noch werden irgendwelche Ein/Ausgabekanäle geschlossen.

SYSTEM RESET (System Reset)

Unterbricht die Programmausführung und aktiviert den Eingabemodus.

Format: SYSTEM RESET
Beispiel: SYSTEM RESET

Beim Drücken der SYSTEM RESET-Taste unterbricht der Computer jede Aufgabe, die er gerade erledigt, ganz gleich, worum es sich dabei handelt. Dann läuft ein Initialisierungsprozeß ab. Dabei wird auf Bogenmaß umgeschaltet, der Wert der Farbregister entsprechend den Voreinstellungen geändert (siehe Tabelle 11-2) und die Grafikbetriebsart 0 aktiviert. Der Bildschirm wird vom Bildschirmeditor kontrolliert. Die Bildschirmgrenzen und die Tabulator-Stops werden auf ihre Normalwerte gesetzt. Jegliche Tonerzeugung wird abgeschaltet und alle Ein/Ausgabekanäle bis auf Kanal 0 sofort geschlossen. Dabei können unter Umständen Daten verloren gehen.
Wenn nach dem Drücken der SYSTEM RESET-Taste ein CONT-Befehl ausgeführt wird, dann wird die Programmausführung mit der Zeile fortgesetzt, die direkt auf die Zeile folgt, in der die Unterbrechung auftrat. Es ist allerdings sehr unwahrscheinlich, daß ein auf diese Art fortgesetztes Programm noch ordnungsgemäß arbeitet.

TRAP (T.)

Nach der Ausführung eines TRAP-Befehls erfolgt, sobald zu einem späteren Zeitpunkt im Programm ein Fehler auftritt, ein Sprung zu der angegebenen Zeile.

Format: TRAP *ProZeil*
Beispiele: TRAP 20000
T. ERRZEIL

Wenn nach der Ausführung eines TRAP-Befehls ein Fehler auftritt, wird nicht, wie sonst üblich, das Programm unterbrochen und eine Fehlermeldung ausgegeben, sondern stattdessen die Programmausführung mit der Zeile fortgesetzt, die die angegebene Zeilennummer (*ProZeil*) trägt. Dort muß der aufgetretene Fehler dann bearbeitet oder behoben werden. Der TRAP-Befehl muß vor dem Auftreten eines Fehlers ausgeführt werden. Anderenfalls wird die übliche Fehlermeldung ausgegeben.
Jeder Fehler ist durch eine Kodezahl gekennzeichnet. Die Kodezahl des zuletzt aufgetretenen Fehlers ist in der Speicherzelle 195 gespeichert. Mit Hilfe des PEEK-Befehls können Sie diese Kodezahl lesen. In Teil A des Anhanges sind die Fehlerkodes und die dazugehörigen Fehlerbeschreibungen aufgeführt.
Die Zeilennummer, in der der Fehler auftrat, kann mit Hilfe des Ausdruckes 256 * PEEK(187) + PEEK(186) bestimmt werden. Wenn der Fehler im Eingabemodus auftrat, ist diese Zeilennummer bedeutungslos.
Wenn nach der Ausführung eines TRAP-Befehls ein Fehler auftritt, dann "vergißt" der Computer diesen TRAP-Befehl und gibt beim Auftreten eines weiteren Fehlers, wie üblich, auf dem Bildschirm eine Fehlermeldung aus. Falls das nicht geschehen soll, muß nach dem Auftreten eines Fehlers erneut ein TRAP-Befehl ausgeführt werden. Wenn bei einem TRAP-Befehl der Wert von *ProZeil* zwischen 32768 und 65535 liegt, dann wird auf den normalen Betrieb umgeschaltet, das heißt, es wird beim Auftreten eines Fehlers auf dem Bildschirm eine Fehlermeldung ausgegeben.

XIO (X.)

Wird allgemein für Ein- und Ausgabe verwendet.

Format: XIO *Bef, #Kan, NumAusdr*$_1$, *NumAusdr*$_2$, *Ger*
Beispiele: XIO 18, #6, 0, 0, "S:"
X. LOCK, #3, 0, 0,FILES1$

Mit Hilfe des XIO-Befehls können sehr viele verschiedene Ein- oder Ausgabeoperationen durchgeführt werden. In Tabelle 11-14 sind alle Möglichkeiten aufgeführt. Zur Auswahl von einer dieser Möglichkeiten dient *Bef*. Je nachdem, welche dieser Funktionen ausgeführt werden soll, muß Kanal *Kan* entweder für Eingabe oder für Ausgabe geöffnet sein. Eine Ausnahme ist der XIO 3-Befehl (OPEN). Hier darf Kanal *Kan* nicht bereits geöffnet sein. Mit Hilfe von *Ger* wird festgelegt, welches Gerät angesprochen wird. Die beiden numerischen Ausdrücke (*NumAusdr*) enthalten zusätzliche Informationen. Wozu sie verwendet werden, hängt davon ab, welche Funktion gewählt wurde. Es müssen grundsätzlich alle Parameter angegeben werden, auch dann, wenn nicht alle benutzt werden. In den Tabellen 11-14 bis 11-17 ist aufgeführt, welche Werte bei den verschiedenen Funktionen für die Parameter eingesetzt werden können und was damit beeinflußt wird. Nichtganzzahlige Zahlen werden gerundet.

Tabelle 11-14: Die verschiedenen XIO-Befehle

Funktion	Wert von *BEF*	Entsprechender BASIC-Befehl	Wert von $NumAusdr_1$	Wert von $NumAusdr_2$
Allgemeine Funktion:				
Kanal öffnen	3	OPEN	Tabelle 11-8	Tabelle 11-9
Eine Zeile einlesen	5	INPUT	0	0
Ein Zeichen einlesen	7	GET	0	0
Eine Zeile ausgeben	9	PRINT	0	0
Ein Zeichen ausgeben	11	PUT	0	0
Kanal schließen	12	CLOSE	0	0
Ergebnis der letzten Ein/Ausgabeoperation	13	STATUS	0	0
Grafikfunktionen:				
Zeichnen einer Linie[1]	17	DRAWTO	0	0
Ausfüllen einer Fläche[2]	18	Nicht vorhanden	0	0
Zum Ansprechen der Diskettenstation:[3]				
Umbenennen eines Files[4]	32	DOS-Menue	0	0
Löschen eines Files[5]	33	DOS-Menue	0	0
Ein File gegen Überschreiben schützen[5]	35	DOS-Menue	0	0
Schreibschutz eines Files entfernen[5]	36	DOS-Menue	0	0
Zeiger eines Files positionieren[5,6]	37	POINT[6]	0	0
Position des Zeigers eines Files feststellen[5,6]	38	NOTE[6]	0	0
Diskette formatieren[5]	254	DOS-Menue	0	0
Zum Ansprechen einer RS-232-Schnittstelle:[7]				
Teil eines Blockes ausgeben	32	Nicht vorhanden	0	0
Ein- und Ausschalten von DTR, RTS, XMT	34	Nicht vorhanden	Tabelle 11-15	0
Festlegung der Baudrate, Wortgröße usw.	36	Nicht vorhanden	Tabelle 11-16	Tabelle 11-16
Festlegung der Zeichenübersetzung	38	Nicht vorhanden	Tabelle 11-17	ATASCII-Kode
Gleichzeitige Ein- und Ausgabe	40	Nicht vorhanden	0	0

Anmerkungen:

[1] Bewegen Sie den Cursor vor der Ausführung eines XIO 17-Befehls mit Hilfe eines Position-Befehls an den Anfangspunkt der zu zeichnenden Linie.

[2] Zeichnen Sie vor der Ausführung eines XIO 18-Befehls einige Linien zur Begrenzung der zu füllenden Fläche und bestimmen Sie mit Hilfe des Befehls POKE 765,*NumAusdr*, mit welcher Farbe diese Fläche gefüllt werden soll.

[3] Das Disk Operating System muß sich im Speicher befinden.

[4] Mit Hilfe von *Ger* wird festgelegt, welches File umbenannt werden soll (XIO 32).

[5] Mit Hilfe von *Ger* wird festgelegt, welches File angesprochen wird. Es spielt keine Rolle, für welches File Kanal *Kan* geöffnet ist.

[6] Kann nur in Verbindung mit DOS 2.0S benutzt werden.

[7] Das Programm zur Bedienung der RS-232-Schnittstellen muß beim Einschalten des Computers gebootet worden sein.

Tabelle 11-15: Funktion von *NumAusdr*$_1$ beim XIO 34-Befehl

Um den Wert zu erhalten, der für *NumAusdr*$_1$ eingesetzt werden muß, entnehmen Sie jeder Spalte eine Zahl und addieren Sie sie				**Auswirkungen des XIO 34-Befehls beim Einsetzen verschiedener Werte für *NumAusdr*$_1$**			
	DTR	**RTS**	**XMT**	**DTR**	**RTS**	**XMT**	**Wert**
Keine Veränderung	0	0	0	Aus	Aus	0	162
Ausschalten (XMT geht auf 0)	128	32	2	Aus	Aus	1	163
				Aus	Ein	0	178
				Aus	Ein	1	179
Einschalten (XMT geht auf 1)	192	48	3	Ein	Aus	0	226
				Ein	Aus	1	227
				Ein	Ein	0	242
				Ein	Ein	1	243

Tabelle 11-16: Funktion von *NumAusdr*$_1$ und *NumAusdr*$_2$ beim XIO 36-Befehl

***NumAusdr*$_1$ (Entnehmen Sie jeder Spalte eine Zahl und addieren Sie sie)**						***NumAusdr*$_2$**			
Stop-Bits	**Wert**	**Wortgröße**	**Wert**	**Baudrate**	**Wert**	**DSR**	**CTS**	**CRX**	**Wert**
1	0	8 Bits	0	300	0	Nein	Nein	Nein	0
2	128	7 Bits	16	45.5	1	Nein	Nein	Ja	1
		6 Bits	32	50	2	Nein	Ja	Nein	2
		5 Bits	48	56.875	3	Nein	Ja	Ja	3
				75	4	Ja	Nein	Nein	4
				110	5	Ja	Nein	Ja	5
				134.5	6	Ja	Ja	Nein	6
				150	7	Ja	Ja	Ja	7
				300	8				
				600	9				
				1200	10				
				1800	11				
				2400	12				
				4800	13				
				9600	14				
				9600	15				

Tabelle 11-17: Funktion von *NumAusdr*$_1$ beim XIO 38-Befehl

Entnehmen Sie jeder Spalte einen Wert und addieren Sie sie							
Einfügen eines Zeilenvorschubs	**Wert**	**Übersetzung vom ATASCII- in den ASCII-Kode**	**Wert**	**Paritätsprüfung beim Einlesen**	**Wert**	**Parität bei der Ausgabe**	**Wert**
Nein	0	Teilweise	0	Keine	0	Wie vorher	0
Ja*	64	Ja	16	Ungerade**	4	Ungerade	1
		Nein	32	Gerade**	8	Gerade	2
				Keine**	12	Bit eingeschaltet	3

* Nach jedem RETURN-Zeichen (ATASCII-EOL) wird ein Zeichen zur Erzeugung eines Zeilenvorschubs eingefügt.

** Nach der Paritätsprüfung wird das Paritätsbit gelöscht.

Die BASIC-Funktionen

Im nun folgenden Abschnitt werden in alphabetischer Reihenfolge die BASIC-Funktionen beschrieben. Die äußere Form (Formatangabe usw.) entspricht der des vorigen Abschnittes.

ABS

Berechnet den Absolutwert einer Zahl.

Format: ABS(*NumAusdr*)
Beispiel: IF A = ABS(A) THEN PR. "POSITIV"

ADR

Berechnet die Adresse, ab der eine alphanumerische Konstante oder Variable im Speicher abgelegt ist.

Format: ADR(*Alpha*)
Beispiel: A = USR(ADR(TEILR$))

Ein in Maschinensprache geschriebenes Programm kann in einer alphanumerischen Variablen abgelegt werden. Mit Hilfe der ADR-Funktion wird dann die Anfangadresse dieser Variablen und damit die Anfangadresse des in dieser Variablen gespeicherten Programmes berechnet. Anschließend wird diese Adresse im Zusammenhang mit der USR-Funktion dazu benutzt, um das Maschinenprogramm zu starten.

ASC

Berechnet den ATASCII-Kode eines Zeichens.

Format: ASC(*Alpha*)
Beispiel: IF ASC(ANTWORT$) < 78 THEN 990

Wenn *Alpha* länger als ein Zeichen ist, dann wird der ATASCII-Kode des ersten Zeichens berechnet. Falls *Alpha* gar kein Zeichen enthält, hat die berechnete Kodezahl einen Wert von 44. In Teil D des Anhanges sind alle ATASCII-Zeichen mit den dazugehörigen Kodezahlen aufgeführt.

ATN

Berechnet den Arcustangens einer Zahl.

Format: ATN(*NumAusdr*)
Beispiel: PRINT ATN(T)

Wenn Sie nicht vorher mit Hilfe des DEG-Befehls auf Grad umgeschaltet haben, wird die Berechnung im Bogenmaß durchgeführt. Der berechnete Winkel liegt zwischen $-\pi/2$ und $+\pi/2$ beziehungsweise zwischen –180 und +180 Grad.

CHR$

Berechnet zu einer gegebenen ATASCII-Kodezahl das entsprechende Zeichen.

Format: CHR$(*NumAusdr*)
Beispiel: PRINT CHR$(65)

Wenn der Wert von *NumAusdr* (entspricht der gegebenen ATASCII-Kodezahl) nicht ganzzahlig ist, wird er gerundet. In Teil D des Anhanges sind alle ATASCII-Zeichen mit den dazugehörigen Kodezahlen aufgeführt. Der Wert von *NumAusdr* sollte zwischen 0 und 255 liegen. Es tritt jedoch kein Fehler auf, wenn er größer ist (maximal 65535). In diesem Fall wird solange 256 subtrahiert, bis das Ergebnis zwischen 0 und 255 liegt.
Wenn mit Hilfe des Kleiner- (<), Größer- (>) oder Ungleich-Zeichens (< >) Vergleiche angestellt werden, dann darf die CHR$-Funktion nur auf einer Seite dieses Vergleichs stehen.

CLOG

Berechnet den Zehnerlogarithmus einer Zahl.

Format: CLOG(*NumAusdr*)
Beispiel: A = B*CLOG(A)

Bei der Benutzung der CLOG-Funktion wird der Zehnerlogarithmus von *NumAusdr* berechnet. Wenn der Wert von *NumAusdr* gleich 0 oder negativ ist, tritt ein Fehler auf.

COS

Berechnet den Kosinus eines Winkels.

Format: COS(*NumAusdr*)
Beispiel: COS(3.1415)

Wenn Sie nicht vorher mit Hilfe des DEG-Befehls auf Grad umgeschaltet haben, wird die Berechnung im Bogenmaß durchgeführt.

EXP

Berechnet einen Wert der Exponentialfunktion.

Format: EXP(*NumAusdr*)
Beispiel: Y = EXP(X)

Die Basis der Exponentialfunktion ist e (2.71828179). Der Exponent entspricht dem Wert von *NumAusdr*.

FRE

Mit Hilfe der FRE-Funktion kann berechnet werden, wieviel Speicherplatz (in Bytes) im Moment noch frei ist.

Format: FRE(*NumAusdr*)
Beispiel: ? FRE(0)

Der Speicherplatz, der nicht bereits durch das Operating System, das Disk Operating System, die Display List, den Bildschirmspeicher oder ein BASIC-Programm mit den dazugehörigen Daten belegt ist, steht Ihnen zur freien Verfügung.
NumAusdr wird zur Berechnung des freien Speicherplatzes zwar nicht benutzt, aber es triit ein Fehler auf, wenn *NumAusdr* nicht angegeben wird oder der Absolutwert von *NumAusdr* größer als 9,99999999E+98 ist.

INT

Mit Hilfe der INT-Funktion kann eine Zahl gerundet werden.

Format: INT(*NumAusdr*)
Beispiel: A = (INT(A/2)*100+5)/100

Der Wert von *NumAusdr* wird auf die nächstkleinere ganze Zahl abgerundet.

LEN

Berechnet die Länge einer alpanumerischen Variablen.

Format: LEN(*Alpha*)
Beispiel: A$(LEN(A$)+1)= B$

Bei Benutzung der LEN-Funktion werden alle Zeichen gezählt, die in *Alpha* enthalten sind (einschließlich der Zeichen, die nicht auf dem Bildschirm dargestellt werden können sowie der Leerzeichen).

LOG

Berechnet den natürlichen Logarithmus einer Zahl.

Format: LOG(*NumAusdr*)
Beispiel: A = B*LOG(A)

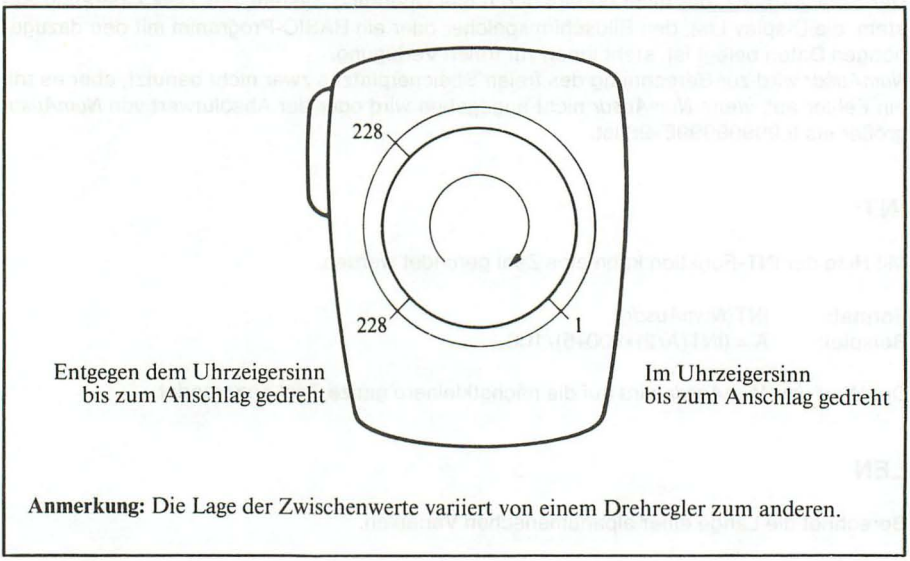
Bei der Benutzung der LOG-Funktion wird der natürliche Logarithmus (Basis e) von *NumAusdr* berechnet. Wenn der Wert von *NumAusdr* gleich 0 oder negativ ist, tritt ein Fehler auf.

PADDLE

Mit Hilfe der PADDLE-Funktion kann die momentane Stellung des angegebenen Drehreglers bestimmt werden.

Format: PADDLE(*NumAusdr*)
Beispiel: PLOT PADDLE(0)/6, PADDLE(1)/12

Je nachdem, in welcher Stellung sich der mit Hilfe von *NumAusdr* ausgewählte Drehregler befindet (siehe Bild 11-2) beziehungsweise wie groß ein an Stelle des Drehreglers angeschlossener Widerstand ist, liefert die PADDLE-Funktion einen Wert, der zwischen 1 und 228 liegt. Die Drehregler sind von 0 bis 7 numeriert. Wenn der Wert von *NumAusdr* kleiner als 0 oder größer als 255 ist, tritt ein Fehler auf. Wenn er zwischen 8 und 255 liegt, liefert die PADDLE-Funktion kein brauchbares Ergebnis.



Anmerkung: Die Lage der Zwischenwerte variiert von einem Drehregler zum anderen.

Bild 11-2: Das Ergebnis der PADDLE-Funktion

PEEK

Mit Hilfe der PEEK-Funktion kann der Inhalt einer Speicherzelle gelesen werden.

Format: PEEK(*SpAdr*)
Beispiel: LGRENZ = PEEK(82)

Die PEEK-Funktion berechnet den Dezimalwert der in der Speicherzelle *SpAdr* gespeicherten binären Information. Einige oft benutzte Speicherzellen sind in Teil G des Anhanges aufgeführt.

PTRIG

Mit Hilfe der PTRIG-Funktion kann festgestellt werden, ob der Taster an dem angegebenen Drehregler gedrückt ist.

Format: PTRIG(*NumAusdr*)
Beispiel: IF PTRIG(1) = 0 THEN PRINT "BUMM!"

Wenn das Ergebnis der PTRIG-Funktion 0 ist, dann ist der Taster des mit Hilfe von *NumAusdr* ausgewählten Drehreglers gedrückt. Wenn er nicht gedrückt ist, dann liefert die PTRIG-Funktion als Ergebnis eine 1. Die Drehregler sind von 0 bis 7 numeriert. Wenn der Wert von *NumAusdr* kleiner als 0 oder größer als 255 ist, tritt ein Fehler auf. Wenn er zwischen 8 und 255 liegt, liefert die PTRIG-Funktion kein brauchbares Ergebnis.

RND

Berechnet eine Zufallszahl.

Format: RND(*NumAusdr*)
Beispiel: IF RND(0) < 0,3 THEN TOTO = 1

Das Ergebnis der RND-Funktion ist eine Zahl, deren Wert größer oder gleich 0 und kleiner als 1 ist. *NumAusdr* hat zwar keinen Einfluß auf das Ergebnis der RND-Funktion, muß aber trotzdem angegeben werden.

SGN

Mit Hilfe der SGN-Funktion kann festgestellt werden, ob eine Zahl positiv, negativ oder gleich 0 ist.

Format: SGN(*NumAusdr*)
Beispiel: IF SGN(A) = –1 THEN PRINT "NEGATIV"

Das Ergebnis der SGN-Funktion ist +1, wenn der Wert von *NumAusdr* positiv, –1, wenn er negativ und 0, wenn er 0 ist.

SIN

Berechnet den Sinus eines Winkels.

Format: SIN(*NumAusdr*)
Beispiel: SIN (WINKEL)

Wenn Sie nicht vorher mit Hilfe des DEG-Befehls auf Grad umgeschaltet haben, wird die Berechnung im Bogenmaß durchgeführt.

SQR

Berechnet die Quadratwurzel einer Zahl.

Format: SQR(*NumAusdr*)
Beispiel: HYPOT = SQR(KAT1 ^ 2 + KAT2 ^ 2)

Wenn der Wert von *NumAusdr* negativ ist, tritt ein Fehler auf. SQR (*NumAusdr*) wird schneller berechnet als *(NumAusdr) ^ (0,5).*

STICK

Mit Hilfe der STICK-Funktion kann die momentane Stellung des angegebenen Steuerknüppels bestimmt werden.

Format: STICK(*NumAusdr*)
Beispiel: IF STICK(0) = 14 THEN ZEIL = ZEIL – 1

Je nachdem, in welcher Stellung sich der mit Hilfe von *NumAusdr* ausgewählte Steuerknüppel befindet (siehe Bild 11-3), liefert die STICK-Funktion einen Wert, der zwischen 0 und 15 liegt. Die Steuerknüppel sind von 0 bis 3 numeriert. Wenn der Wert von *NumAusdr* kleiner als 0 oder größer als 255 ist, tritt ein Fehler auf. Wenn er zwischen 4 und 255 liegt, liefert die STICK-Funktion kein brauchbares Ergebnis.

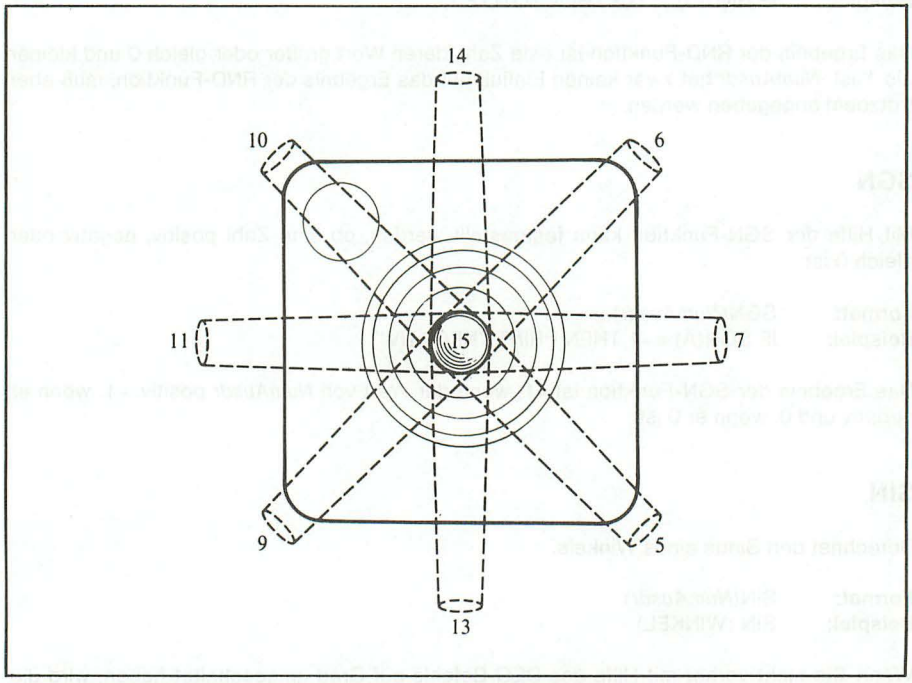


Bild 11-3: Die möglichen Stellungen des Steuerknüppels mit den dazugehörigen Ergebnissen der STICK-Funktion

STRIG

Mit Hilfe der STRIG-Funktion kann festgestellt werden, ob der Taster an dem angegebenen Steuerknüppel gedrückt ist.

Format: STRIG(*NumAusdr*)
Beispiel: IF STRIG(1) = 0 THEN PRINT "BUMM!"

Wenn das Ergebnis der STRIG-Funktion 0 ist, dann ist der Taster des mit Hilfe von *NumAusdr* ausgewählten Steuerknüppels gedrückt. Wenn er nicht gedrückt ist, dann liefert die STRIG-Funktion als Ergebnis eine 1. Die Steuerknüppel sind von 0 bis 3 numeriert. Wenn der Wert von *NumAusdr* kleiner als 0 oder größer als 255 ist, tritt ein Fehler auf. Wenn er zwischen 4 und 255 liegt, liefert die STRIG-Funktion kein brauchbares Ergebnis.

STR$

Wandelt einen numerischen Wert in einen alphanumerischen Wert um.

Format: STR$(*NumAusdr*)
Beispiel: PLZ$ = STR$(PLZ)

Das Ergebnis der STR$-Funktion sieht genauso aus, als wenn *NumAusdr* mit Hilfe des PRINT-Befehls auf dem Bildschirm dargestellt worden wäre. Der Wert von *NumAusdr* wird in ATASCII-Zeichen umgewandelt. Es gilt zum Beispiel STR$(2/3) = "0.6666666666" oder STR$(12300000000) = 1.23E + 10". Wenn der Absolutwert von *NumAusdr* größer als 9,99999999EE + 98 ist, tritt ein Fehler auf.
Wenn mit Hilfe des Kleiner- (<), Größer- (>) oder Ungleich-Zeichens (< >) Vergleiche angestellt werden, dann darf die STR$-Funktion nur auf einer Seite dieses Vergleiches stehen.

USR

Dient zum Aufruf eines Maschinenprogrammes. Es ist möglich, an dieses Maschinenprogramm einige Werte zur weiteren Verarbeitung zu übergeben.

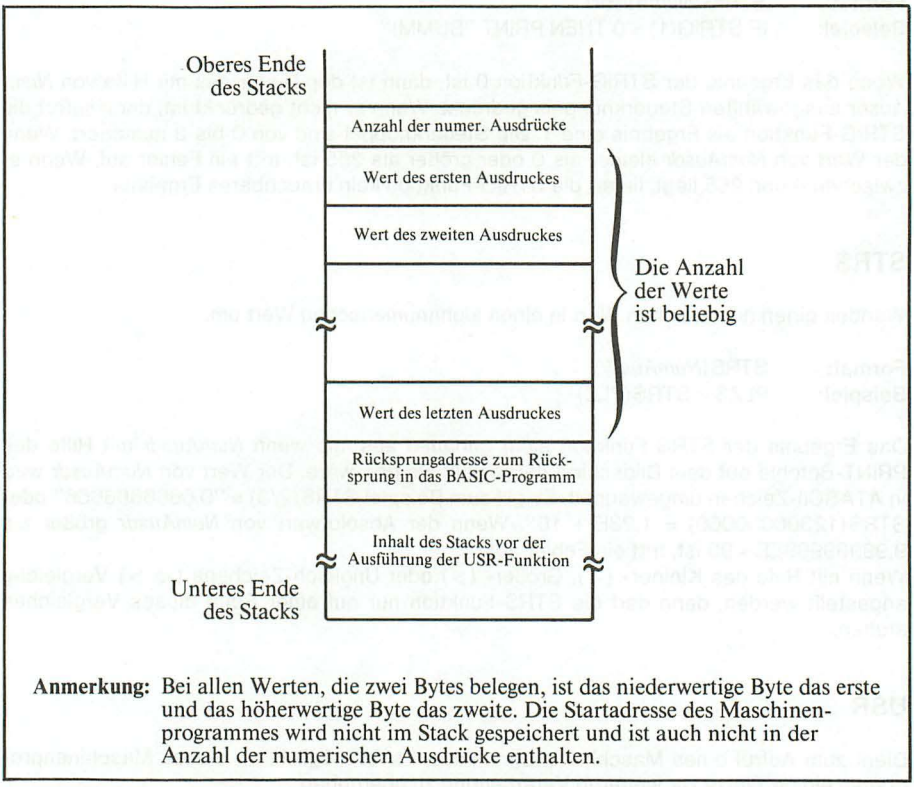
Format: USR(*SpAdr* [,*NumAusdr*...])
Beispiel: A = USR(1536,ADR(A$),ADR(B$))

Bei der Ausführung einer USR-Funktion speichert das BASIC zunächst eine Rücksprungadresse im Stack und ruft dann das Maschinenprogramm auf. *SpAdr* ist die Startadresse des Programmes. Natürlich muß sich das Maschinenprogramm bereits im Speicher befinden.

Die Argumente der USR-Funktion

Der Wert von jedem der bei einer USR-Funktion angegebenen numerischen Ausdrücke (*NumAusdr*) muß zwischen 0 und 65535 liegen. Diese Werte übergibt das BASIC bei der Ausführung der USR-Funktion über den Stack an das Maschinenprogramm. Von hinten beginnend, wird der Wert von jedem numerischen Ausdruck berechnet, falls notwendig, gerundet und in Form einer zwei-Byte-Hexadezimalzahl im Stack gespeichert. Nachdem auf diese Weise alle Ausdrücke bearbeitet worden sind, wird zusätzlich die Anzahl der numerischen Ausdrücke mit Hilfe einer ein-Byte-Hexadezimalzahl im Stack gespeichert (sie-

he Bild 11-4). Das Maschinenprogramm muß diese Werte dem Stack entnehmen (zum Beispiel mit Hilfe von PLA-Befehlen), bevor ins BASIC zurückgesprungen wird. Auch wenn bei einem USR-Befehl kein numerischer Ausdruck angegeben wird, speichert das BASIC die Anzahl der numerischen Ausdrücke (in diesem Fall 0) im Stack. Das Maschinenprogramm muß dem Stack also mindestens ein Byte entnehmen.



Anmerkung: Bei allen Werten, die zwei Bytes belegen, ist das niederwertige Byte das erste und das höherwertige Byte das zweite. Die Startadresse des Maschinenprogrammes wird nicht im Stack gespeichert und ist auch nicht in der Anzahl der numerischen Ausdrücke enthalten.

Bild 11-4: Benutzung des Stack durch die USR-Funktion

Ergebnis der USR-Funktion

Mit Hilfe der Speicherzellen 212 und 213 kann eine zwei-Byte-Hexadezimalzahl an das BASIC-Programm übergeben werden. Das niederwertige Byte muß in Speicherzelle 212 und das höherwertige in Speicherzelle 213 gespeichert werden. Nach dem Rücksprung ins BASIC-Programm wird der Inhalt dieser Speicherzellen in eine Zahl umgewandelt, deren Wert zwischen 0 und 65535 liegt.

Rücksprung ins BASIC-Programm

Sobald im Maschinenprogramm ein RTS-Befehl auftritt, der dem Stack die bei der Ausführung des USR-Befehls dort gespeicherte Rücksprungadresse (siehe Bild 11-4) entnimmt, erfolgt ein Rücksprung ins BASIC-Programm. Diese Tatsache verdeutlicht auch, warum dem Stack vor dem Rücksprung ins BASIC-Programm alle Werte, die an das Maschinenprogramm übergeben wurden, entnommen werden müssen.

VAL

Wandelt einen alphanumerischen Wert in einen numerischen Wert um.

Format: VAL(*Alpha*)
Beispiel: SUM = VAL(A$) + VAL(B$)

Mit Hilfe der Val-Funktion wird die in *Alpha* in alphanumerischer Form gespeicherte Zahl in einen numerischen Wert umgewandelt. Wenn es sich bei dem ersten Zeichen von *Alpha* nicht um ein zur Darstellung von numerischen Werten verwendetes Zeichen handelt, tritt ein Fehler auf. Anderenfalls wird *Alpha* Zeichen für Zeichen umgewandelt, bis ein Zeichen auftritt, das nicht umgewandelt werden kann. In Bild 11-5 ist aufgeführt, wie *Alpha* aussehen muß, damit eine Umwandlung möglich ist.
Wenn die in *Alpha* gespeicherte Zahl zu groß oder zu klein ist (zum Beispiel 1E99), dann tritt ein Fehler auf.

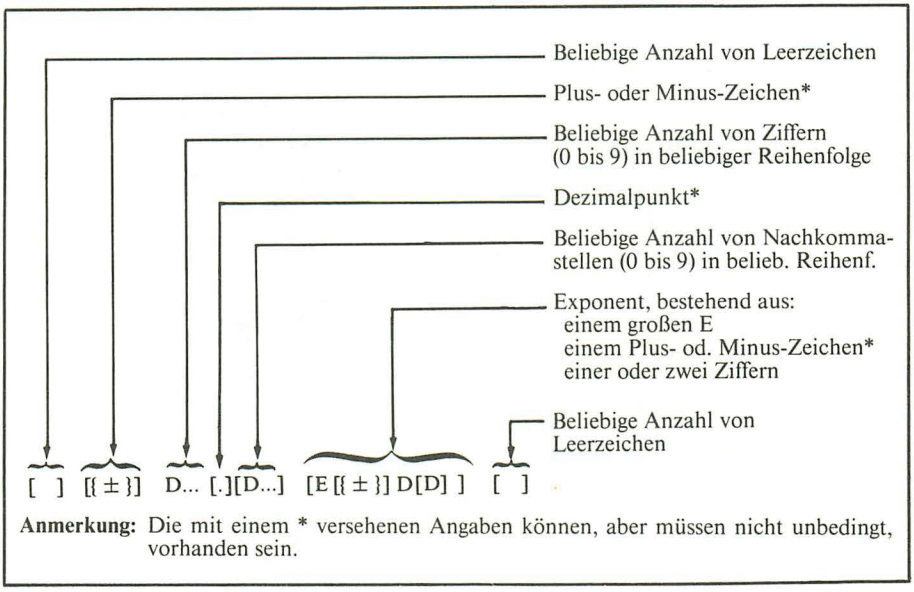


Bild 11-5: So muß *Alpha* aussehen, damit eine Umwandlung mit Hilfe der VAL-Funktion möglich ist.

Anhang A

FEHLERMELDUNGEN UND ERKLÄRUNGEN DAZU

Beim ATARI-Computer wird, wenn ein Fehler auftritt, eine Kodezahl auf dem Bildschirm ausgegeben. Die Bedeutung dieser Kodezahlen ist im folgenden Abschnitt erläutert. Hinter jeder Fehlerbezeichnung steht in Klammern die englische Bezeichnung dieses Fehlers, so wie sie auch in den ATARI-Publikationen verwendet wird.

2 NICHT GENUG SPEICHERPLATZ VORHANDEN (MEMORY INSUFFICIENT)

Es ist nicht genug Speicherplatz zur Speicherung des Programmes oder der Variablen vorhanden, oder es sind zu viele FOR-NEXT-Schleifen oder Unterprogramme ineinander verschachtelt worden.

3 FALSCHER WERT (BAD VALUE)

Ein numerischer Wert ist zu groß, zu klein oder an einer Stelle, wo er positiv sein sollte, negativ.

4 ZU VIELE VARIABLEN (TOO MANY VARIABLES)

In einem Programm dürfen nicht mehr als 128 verschiedene Variablen verwendet werden. Diese Grenze wird unter Umständen auch deshalb überschritten, weil zu der Gesamtzahl der Variablen auch einige früher benutzte Variablen zählen, die inzwischen jedoch nicht mehr gebraucht werden (siehe Bild 11-1).

5 LÄNGE EINER ALPHANUMERISCHEN VARIABLEN ÜBERSCHRITTEN (STRING LENGTH ERROR)

Es ist ein Teil einer alphanumerischen Variablen angesprochen worden, für den diese Variable nicht dimensioniert wurde.

6 ZU WENIG DATA-WERTE (OUT OF DATA ERROR)

Es wurde versucht, mit Hilfe von READ-Befehlen mehr Werte zu lesen als bei DATA-Befehlen angegeben waren.

7 ZAHLENWERT GRÖSSER ALS 32767
(NUMBER GREATER THAN 32767)

Ein numerischer Wert ist an einer Stelle, wo er es nicht sein dürfte, größer als 32767 oder kleiner als 0.

8 INPUT-FEHLER
(INPUT STATEMENT ERROR)

Bei der Ausführung eines INPUT-Befehls wurde festgestellt, daß der angegebene Variablentyp und der Wert, der dieser Variablen zugewiesen werden soll, nicht zueinander passen. Eine numerische Variable kann zum Beispiel keine Buchstaben, Satzzeichen oder Grafikzeichen enthalten.

9 DIMENSIONIERUNGSFEHLER
(ARRAY OR STRING DIMENSION ERROR)

Ein DIM-Befehl enthält ein Feld oder eine alphanumerische Variable, die bereits dimensioniert wurde oder ein Feld, das mehr als 32767 Bytes belegt. Oder es wurde versucht, auf ein noch nicht dimensioniertes Feld oder auf eine noch nicht dimensionierte alphanumerische Variable zuzugreifen.

10 ZU KOMPLEXER AUSDRUCK
(ARGUMENT STACK OVERFLOW)

Ein Ausdruck enthält zuviele Klammern oder zuviele ineinander verschachtelte Funktionen.

11 ZAHL ZU GROSS
(FLOATING POINT OVERFLOW/UNDERFLOW ERROR)

Es wurde versucht, durch 0 zu teilen, oder es ist eine Zahl aufgetreten, deren Absolutwert größer ist als $9.99999999 \times 10^{97}$.

12 ZEILE NICHT GEFUNDEN
(LINE NOT FOUND)

Bei einem GOSUB-, GOTO-, IF-THEN-, ON-GOSUB- oder ON-GOTO-Befehl steht eine Zeilennummer, die im Programm nicht vorkommt.

13 FOR-BEFEHL FEHLT
(NO MATCHING FOR STATEMENT)

Es wurde ein NEXT-Befehl gefunden, zu dem der entsprechende FOR-Befehl fehlt. Möglicherweise überschneiden sich zwei FOR-NEXT-Schleifen, oder der letzte FOR-Befehl wurde dadurch unwirksam gemacht, daß innerhalb einer FOR-NEXT-Schleife (aber nicht nach einem GOSUB-Befehl!!!) ein POP-Befehl ausgeführt wurde.

14 ZEILE ZU LANG
(LINE TO LONG)

Ein Befehl ist zu komplex oder geht über das Ende einer logischen Zeile hinaus.

15 ZEILE MIT GOSUB- ODER FOR-BEFEHL WURDE GELOESCHT (GOSUB OR FOR LINE DELETED)

Bei der Ausführung eines RETURN- oder NEXT-Befehls konnte die Zeile, in der sich der entsprechende GOSUB- oder FOR-Befehl befand, nicht mehr gefunden werden.

16 GOSUB-BEFEHL FEHLT (RETURN ERROR)

Es ist ein RETURN-Befehl gefunden worden, obwohl noch kein GOSUB-Befehl ausgeführt wurde.

17 BEFEHL NICHT AUSFÜHRBAR (GARBAGE ERROR)

Ein Befehl wurde durch einen Fehler im Speicher, einen POKE-Befehl oder ein Maschinenprogramm so verändert, daß er nicht mehr erkannt und somit nicht ausführbar ist.

18 FALSCHES ZEICHEN (INVALID STRING CHARACTER)

Es wurde versucht, eine alphanumerische Variable mit Hilfe der VAL-Funktion in einen numerischen Wert umzuwandeln, obwohl in dieser Variablen Zeichen enthalten sind, die nicht umgewandelt werden können.

19 PROGRAMM IST ZU LANG (LOAD PROGRAM TOO LONG)

Es wurde versucht, ein Programm zu laden, das mehr Speicherplatz belegt als im Computer vorhanden ist.

20 FALSCHE KANALNUMMER (BAD CHANNEL NUMBER)

Es wurde versucht, Kanal 0 zu benutzen, oder es wurde eine Kanalnummer angegeben, die größer als 7 ist.

21 LADEN MIT LOAD NICHT MÖGLICH (LOAD FILE ERROR)

Es wurde versucht, Daten oder ein unter Verwendung des CSAVE- oder LIST-Befehls gespeichertes Programm mit Hilfe des LOAD-Befehls zu laden.

128 ABBRUCH DURCH BREAK-TASTE (BREAK ABORT)

Während einer Ein/Ausgabeoperation wurde die BREAK-Taste gedrückt.

129 KANAL BEREITS OFFEN (IOCB ALREADY OPEN)

Es wurde versucht, einen Kanal zu öffnen, der bereits offen war. Zur Ausführung der Grafikbefehle wird zum Beispiel Kanal 6 und zur Ausführung einiger anderer Befehle Kanal 7 verwendet. Beim Auftreten dieses Fehlers wird der Kanal, der den Fehler verursacht hat, unter Umständen automatisch geschlossen.

130 UNBEKANNTES GERÄT (NONEXISTENT DEVICE)

Es wurde versucht, auf ein unbekanntes Gerät zuzugreifen. In Tabelle 11-10 sind alle Gerätenamen aufgeführt. Beachten Sie, daß die seriellen Schnittstellen (Gerätename R:, R1:, R2:, R3: oder R4:) nur dann benutzt werden können, wenn beim Einschalten des Computers ein Programm zur Bedienung der RS-232-Schnittstellen gebootet wurde (siehe Kapitel 2).

131 KANAL NUR FÜR AUSGABE GEÖFFNET (IOCB WRITE ONLY)

Es wurde ein GET- oder INPUT-Befehl im Zusammenhang mit einem Ein/Ausgabekanal verwendet, der nur für Ausgabe geöffnet ist.

132 XIO-BEFEHL FEHLERHAFT (INVALID COMMAND)

Bei der Ausführung eines XIO-Befehls ist ein Fehler aufgetreten.

133 KANAL IST NICHT GEÖFFNET (DEVICE OR FILE NOT OPEN)

Es wurde versucht, einen Ein/Ausgabekanal zu benutzen, der noch nicht geöffnet war.

134 FALSCHE KANALNUMMER (BAD IOCB NUMBER)

Es können nur Ein/Ausgabekanäle mit den Nummern 1, 2, 3, 4, 5, 6 und 7 verwendet werden.

135 KANAL NUR FÜR EINGABE GEÖFFNET (IOCB READ ONLY)

Es wurde ein PUT- oder PRINT-Befehl im Zusammenhang mit einem Ein/Ausgabekanal verwendet, der nur für Eingabe geöffnet ist.

136 FILEENDE ERREICHT (EOF)

Es wurde ein End-of-file-Record gelesen, oder es wurde versucht, einen Sektor der Diskette zu lesen, der nicht zu dem geöffneten File gehört.

137 RECORD ZU LANG (TRUNCATED RECORD)

Der Computer konnte einen Record nicht vollständig empfangen, weil er länger als 256 Bytes war.

138 GERÄT NICHT ANSPRECHBAR (DEVICE TIMEOUT)

Das angegebene Gerät konnte nicht angesprochen werden. Vergewissern Sie sich, ob das Gerät eingeschaltet ist, ob alle notwendigen Verbindungen ordnungsgemäß hergestellt sind und ob alle Schalter mit einer "ONLINE"- und einer "LOCAL"-Stellung in der

"ONLINE"-Stellung stehen.

139 GERÄT ARBEITET NICHT EINWANDFREI (DEVICE NAK)

Der Programmrecorder oder die Diskettenstation arbeitet nicht einwandfrei oder kann einen bestimmten Befehl nicht ausführen. Letzteres kann auch beim ATARI-Interfacemodul 850 vorkommen. In diesem Fall handelt es sich meistens um die Übertragung von 5, 6 oder 7 Bit breiten Worten mit einer zu hohen Baudrate. Es kann aber auch passieren, daß ein an das Interfacemodul angeschlossenes Gerät nicht zur Datenübertragung bereit ist.

140 FEHLER AUF DEM SERIELLEN BUS (SERIAL BUS)

Bei der Datenübertragung über den seriellen Bus ist ein Fehler aufgetreten. Möglicherweise ist eine Kassette oder eine Diskette defekt.

141 CURSOR AUSSERHALB DER BILDSCHIRMBEGRENZUNG (CURSOR OUT OF RANGE)

Die Maximalzahl der Zeilen und Spalten variiert von einer Grafikbetriebsart zur anderen (siehe Tabelle 11-7).

142 FORMATFEHLER BEI DER DATENÜBERTRAGUNG ÜBER DEN SERIELLEN BUS (SERIAL BUS DATA FRAME OVERRUN)

Bei der Datenübertragung über den seriellen Bus ist ein Fehler aufgetreten. Möglicherweise ist eine Kassette oder eine Diskette defekt.

143 PRÜFSUMMENFEHLER BEI DER DATENÜBERTRAGUNG ÜBER DEN SERIELLEN BUS (SERIAL BUS DATA FRAME CHECKSUM)

Bei der Datenübertragung über den seriellen Bus ist ein Fehler aufgetreten. Eine Kassette oder Diskette konnte nicht beschrieben oder gelesen werden. Möglicherweise ist die Kassette oder Diskette defekt.

144 DISKETTENFEHLER (DISK ERROR)

Eine Diskette ist schreibgeschützt oder in der Directory befindet sich ein Fehler.

145 DISKETTENSCHREIBFEHLER ODER FEHLER BEI DER BILDSCHIRMDARSTELLUNG (READ-AFTER-WRITE COMPARE ERROR, OR BAD SCREEN MODE HANDLER)

Es wurde ein Unterschied festgestellt zwischen dem, was auf eine Diskette geschrieben werden sollte und dem, was tatsächlich geschrieben wurde, oder es wurde ein Fehler im Zusammenhang mit der Bildschirmdarstellung festgestellt.

146 FUNKTION NICHT AUSFÜHRBAR (FUNCTION NOT IMPLEMENTED)

Es wurde versucht, eine nicht ausführbare Funktion auszuführen, zum Beispiel Daten an die Tastatur auszugeben oder Daten vom Drucker zu lesen.

147 SPEICHERPLATZ REICHT FÜR DIE GEWÄHLTE GRAFIKBETRIEBSART NICHT AUS (INSUFFICIENT RAM)

Für die verschiedenen Grafikbetriebsarten wird unterschiedlich viel Speicherplatz benötigt (siehe Tabelle 11-7).

150 SERIELLE SCHNITTSTELLE BEREITS BELEGT (PORT ALREADY OPEN)

Es kann immer nur ein Ein/Ausgabekanal für eine bestimmte serielle Schnittstelle geöffnet sein.

151 SCHNITTSTELLE WURDE NICHT FÜR GLEICHZEITIGE EIN/AUSGABE GEÖFFNET (CONCURRENT MODE I/O NOT ENABLED)

Bevor über eine serielle Schnittstelle Daten ein- oder ausgegeben werden können, muß diese Schnittstelle mit Hilfe eines XIO 40-Befehls für gleichzeitige Ein/Ausgabe geöffnet worden sein (siehe Tabelle 11-8 und 11-14).

152 BUFFERFEHLER BEI GLEICHZEITIGER EIN/AUSGABE (ILLEGAL USER-SUPPLIED BUFFER)

Die bezüglich des Buffers gemachten Angaben waren nicht korrekt.

153 GLEICHZEITIGE EIN/AUSGABE AKTIVIERT (ACTIVE CONCURRENT MODE I/O ERROR)

Es wurde versucht, über eine serielle Schnittstelle Daten ein- oder auszugeben, während eine andere Schnittstelle für gleichzeitige Ein/Ausgabe geöffnet war.

154 GLEICHZEITIGE EIN/AUSGABE NICHT AKTIVIERT (CONCURRENT MODE I/O NOT ACTIVE)

Es wurde versucht, eine Ein- oder Ausgabeoperation im Zusammenhang mit einer seriellen Schnittstelle durchzuführen, bei der diese Schnittstelle für gleichzeitige Ein/Ausgabe geöffnet sein muß (was nicht der Fall war).

160 UNBEKANNTE DISKETTENSTATION (DRIVE NUMBER ERROR)

Wenn die Diskettenstation angesprochen wird, können nur die Gerätenamen D:, D1:, D2:, D3: oder D4: verwendet werden.

161 ZU VIELE FILES OFFEN (TOO MANY FILES OPEN)

Normalerweise können nur drei Files gleichzeitig geöffnet sein. Unter DOS 2.0S kann diese Zahl jedoch erhöht werden (siehe Kapitel 7).

162 KEIN SPEICHERPLATZ MEHR AUF DER DISKETTE (DISK FULL)

Auf der Diskette ist kein Speicherplatz mehr vorhanden; alle Sektoren sind belegt.

163 SYSTEMFEHLER (UNRECOVERABLE SYSTEM DATA I/O ERROR)

Während der Ein- oder Ausgabe von Daten wurde ein Fehler festgestellt, der nicht behoben und dessen Ursache nicht festgestellt werden konnte.

164 FILE/SEKTOR-FEHLER (FILE NUMBER MISMATCH)

Der Zeiger eines Files wurde mit Hilfe eines POINT-Befehls auf einen Sektor gesetzt, der nicht zu dem geöffneten File gehört, oder die Verbindungen zwischen den einzelnen Sektoren sind zerstört.

165 FALSCHER FILENAME (FILE NAME ERROR)

Ein Filename beginnt mit einem Kleinbuchstaben oder enthält unzulässige Zeichen. Es ist auch möglich, daß die Wild-Card-Zeichen nicht richtig benutzt wurden.

166 POINT-BEFEHL FEHLERHAFT (POINT DATA LENGTH ERROR)

Das bei einem POINT-Befehl angegebene Byte existiert nicht.

167 FILE IST SCHREIBGESCHÜTZT (FILE LOCKED)

Schreibgeschützte Files können weder beschrieben oder gelöscht, noch kann der Name dieser Files geändert werden.

168 UNBEKANNTER XIO-BEFEHL (COMMAND INVALID)

Es wurde versucht, einen XIO-Befehl auszuführen, den es nicht gibt.

169 KEIN PLATZ MEHR IN DER DIRECTORY (DIRECTORY FULL)

In der Directory können 64 Filenamen eingetragen werden, unabhängig davon, wieviel Speicherplatz auf der Diskette noch vorhanden ist.

170 FILE NICHT GEFUNDEN (FILE NOT FOUND)

Auf der in der angegebenen Diskettenstation eingelegten Diskette kann kein File mit dem angegebenen Filenamen gefunden werden

171 POINT-BEFEHL NICHT AUSFÜHRBAR (POINT INVALID)

Es wurde versucht, auf einen Sektor zuzugreifen, der nicht zu dem geöffneten File gehört.

Anhang B

DIE STATUSKODES

In diesem Abschnitt sind die Kodezahlen aufgeführt, die bei der Ausführung eines STATUS-Befehls ausgegeben werden können. Daneben ist die Bedeutung dieser Kodezahlen erläutert. Eine ausführliche Beschreibung finden Sie in Teil A des Anhanges.

Kodezahl (dezimal)	Bedeutung
1	Operation erfolgreich und ohne Fehler beendet
3	End-of-File bald erreicht: Mit dem nächsten Lesevorgang werden die letzten Daten des Files gelesen*
128	BREAK-Taste gedrückt
129	Kanal ist bereits geöffnet
130	Unbekanntes Gerät
131	Kanal ist für Ausgabe geöffnet
133	Kanal ist nicht geöffnet
134	Unbekannte Kanalnummer
135	Kanal ist für Eingabe geöffnet
136	Fileende erreicht
137	Record zu lang
138	Gerät nicht ansprechbar
139	Gerät arbeitet nicht einwandfrei
140	Fehler auf dem seriellen Bus
141	Cursor außerhalb der Bildschirmbegrenzung
142	Formatfehler bei der Datenübertragung über den seriellen Bus
143	Prüfsummenfehler bei der Datenübertragung über den seriellen Bus
144	Diskette schreibgeschützt
145	Diskettenschreibfehler oder Fehler bei der Bildschirmdarstellung
146	Funktion nicht ausführbar
147	Speicherplatz reicht für die gewählte Grafikbetriebsart nicht aus
160	Unbekannte Diskettenstation
161	Zu viele Files offen
162	Kein Speicherplatz mehr auf der Diskette vorhanden
163	Systemfehler
164	File/Sektor-Fehler
165	Falscher Filename
166	POINT-Befehl fehlerhaft
167	File ist schreibgeschützt
168	Unbekannter XIO-Befehl
169	Kein Platz mehr in der Directory
170	File nicht gefunden
171	POINT-Befehl nicht ausführbar

* Die Bedeutung dieser Kodezahl entspricht nicht der in Teil A des Anhanges angegebenen Bedeutung.

Anhang C

ABGELEITETE TRIGONOMETRISCHE FUNKTIONEN

Die folgende Zusammenstellung von abgeleiteten trigonometrischen Funktionen ist nicht vollständig. Es sollen hier vielmehr nur die Funktionen aufgeführt werden, die am häufigsten vorkommen. Manche Funktionswerte sind irrational; zum Beispiel kann SEC(X) nicht berechnet werden, wenn COS(X)=0 ist. Wenn solche Funktionen in einem Programm verwendet werden, sollte vor der Berechnung eines Funktionswertes immer eine entsprechende Überprüfung stattfinden.

ARCCOS(X) = -ATN(X/SQR/(-X * X + 1)) + 1,5707633
Berechnet den Arcus Cosinus von X (ABS(X) < 1).

ARCCOT(X) = -ATN(X) + 1.5707633
Berechnet den Arcus Cotangens von X.

ARCCOSH(X) = LOG(X + SQR(X * X - 1))
Berechnet den Arcus Cosinus Hyperbolicus von X (X >= 1).

ARCCOTH(X) = LOG((X + 1)/(X - 1))/2
Berechnet den Arcus Cotangens Hyperbolicus von X (ABS(X) > 1).

ARCCSC(X) = ATN(1/SQR(X * X - 1)) + (SGN(X) - 1) * 1.5707633
Berechnet den Arcus Cosekans von X (ABS(X) > 1).

ARCCSCH(X) = LOG((SGN(X) * SQR(X * X + 1) + 1)/X)
Berechnet den Arcus Cosekans Hyperbolicus von X (X > 0).

ARCSEC(X) = ATN(SQR(X * X - 1)) + (SGN(X) - 1) * 1.5707633
Berechnet den Arcus Sekans von X (ABS(X) >= 1).

ARCSECH(X) = LOG((SQR(-X * X + 1) + 1)/X)
Berechnet den Arcus Sekans Hyperbolicus von X (0 < X <= 1).

ARCSIN(X) = ATN(X/SQR(-X * X + 1))
Berechnet den Arcus Sinus von X (ABS(X) < 1).

ARCSINH(X) = LOG(X + SQR(X * X + 1))
Berechnet den Arcus Sinus Hyperbolicus von X.

ARCTANH(X) = LOG((1 + X)/(1 - X))/2
Berechnet den Arcus Tangens Hyperbolicus von X (ABS(X) < 1).

COSH(X) = (EXP(X) + EXP(–X))/2
Berechnet den Cosinus Hyperbolicus von X.

COT(X) = COS(X)/SIN(X)
Berechnet den Cotangens von X (X <> 0).

COTH(X) = EXP(–X)/(EXP(X) – EXP(–X)) * 2 + 1
Berechnet den Cotangens Hyperbolicus von X (X <> 0).

CSC(X) = 1/SIN(X)
Berechnet den Cosekans von X (X <> 0).

CSCH(X) = 2/(EXP(X) – EXP(–X))
Berechnet den Cosekans Hyperbolicus von X (X <> 0).

$LOG_a(X)$ = LOG(X)/LOG(a)
Berechnet den Logarithmus von X zur Basis a (a > 0, X > 0).

$LOG_{10}(X)$ = LOG(X)/2.30258509
Berechnet den Zehnerlogarithmus von X (X > 0).

$MOD_a(X)$ = INT((X/a – INT(X/a)) * a + 0.05) * SGN(X/a)
Berechnet X modulo a: der Rest, nachdem X durch a dividiert wurde (a <> 0).

SEC(X) = 1/COS(X)
Berechnet den Sekans von X (X <> π/2).

SECH(X) = 2/(EXP(X) + EXP(–X))
Berechnet den Sekans Hyperbolicus von X.

SINH(X) = (EXP(X) – EXP(–X))/2
Berechnet den Sinus Hyperbolicus von X.

TAN(X) = SIN(X)/COS(X)
Berechnet den Tangens von X (X <> 0).

TANH(X) = –EXP(–X)/EXP(X) + EXP(–X)) * 2 + 1
Berechnet den Tangens Hyperbolicus von X.

Anhang D

ATASCII-KODES, ATASCII-ZEICHEN UND DIE FUNKTION DER TASTATUR

In Tabelle D-1 sind alle 256 Zeichen des normalen Zeichensatzes, die in der Grafikbetriebsart 0 auf dem Bildschirm dargestellt werden können, enthalten. Bei jedem Zeichen ist die dazugehörige ATASCII-Kodezahl angegeben. Wenn Sie diese Kodezahl in eine CHR$-Funktion einsetzen, dann wird das entsprechende Zeichen erzeugt. Stattdessen können jedoch auch eine oder mehrere bestimmte Tasten gedrückt werden. Auch diese Information ist in Tabelle D-1 enthalten.
Bei der Betätigung einer oder mehrerer Tasten (wie in Tabelle D-1 angegeben) wird auf jeden Fall die in der Tabelle aufgeführte Kodezahl erzeugt. In der Grafikbetriebsart 0 können auch die entsprechenden Zeichen auf dem Bildschirm dargestellt werden. In anderen Grafikbetriebsarten kann es jedoch vorkommen, daß bei einigen Kodes an Stelle der angegebenen Zeichen andere erzeugt werden (siehe Tabelle 11-4) oder daß keine Zeichen, sondern Grafikpunkte auf dem Bildschirm dargestellt werden.
Einige Kodezahlen dienen der Erzeugung von Controlzeichen. Wenn ein Controlzeichen mit Hilfe eines PRINT-Befehls auf den Bildschirm gebracht wird, wird kein Zeichen dargestellt, sondern stattdessen eine Controlfunktion ausgeführt, zum Beispiel der Cursor nach oben, unten, rechts oder links bewegt. Um ein Controlzeichen mit Hilfe eines PRINT-Befehls auszugeben, muß entweder die entsprechende Kodezahl im Zusammenhang mit der CHR$-Funktion verwendet oder das Controlzeichen in Anführungszeichen nach Betätigung der ESC-Taste angegeben werden (siehe Kapitel 4). Wenn nur die bei einem Controlzeichen angegebenen Tasten gedrückt werden, dann wird die entsprechende Controlfunktion ausgeführt. Falls jedoch vorher die ESC-Taste gedrückt wurde, so wird stattdessen das bei dem ATASCII-Kode des Controlzeichens angegebene Zeichen auf dem Bildschirm dargestellt. Erst wenn der PRINT-Befehl, in dem dieses Zeichen steht, ausgeführt wird, findet die Controlfunktion statt. Das Zeichen repräsentiert also nur die entsprechende Controlfunktion. Wenn jedoch direkt vor einem Controlzeichen ein Zeichen mit dem ATASCII-Kode 27 auf den Bildschirm gebracht wird, dann wird die Controlfunktion nicht ausgeführt, sondern stattdessen das Zeichen, das diese Controlfunktion repräsentiert, auf dem Bildschirm dargestellt. In Tabelle D-1 ist jedes Controlzeichen mit einer Fußnote versehen. Dort wird dann die entsprechende Controlfunktion erläutert.

Kleinbuchstaben

Um Kleinbuchstaben direkt eingeben zu können, muß die Tastatur umgeschaltet werden. Dies geschieht durch Drücken der LOWR-Taste. Diese Umschaltung muß bei allen Zei-

chen vorgenommen werden, bei denen in der Spalte "Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)" (LOWR) steht. Nach dieser Umschaltung kann mit der Tastatur wie mit einer Schreibmaschine groß und klein geschrieben werden. Nach einer erneuten Umschaltung, bei der durch Drücken der CAPS-Taste (SHIFT-LOWR) wieder der Zustand hergestellt worden ist, der auch beim Einschalten des Computers vorhanden war, können nur noch Großbuchstaben geschrieben werden.

Inverse Zeichendarstellung

Fast die Hälfte aller Zeichen sind invers dargestellt. Um diese Zeichen direkt eingeben zu können, muß die Tastatur umgeschaltet werden. Die Umschaltung muß bei allen Zeichen vorgenommen werden, bei denen in der Spalte "Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)" ein ⅄-Zeichen steht. Wenn die ⅄-Taste gedrückt wird, wird auf inverse Zeichendarstellung umgeschaltet. Nachdem sie das zweite Mal gedrückt worden ist, werden alle Zeichen wieder normal dargestellt. Bei jeder Betätigung dieser Taste wird auf die Zeichendarstellung umgeschaltet, die gerade nicht benutzt wird.

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
0		NULL	CTRL-,	11		VT	CTRL-K
1		SOH	CTRL-A	12		FF	CTRL-L
2		STX	CTRL-B	13		CR	CTRL-M
3		ETX	CTRL-C	14		SO	CTRL-N
4		EOT	CTRL-D	15		SI	CTRL-O
5		ENQ	CTRL-E	16		DLE	CTRL-P
6		ACK	CTRL-F	17		DC1	CTRL-Q
7		BEL	CTRL-G	18		DC2	CTRL-R
8		BS	CTRL-H	19		DC3	CTRL-S
9		HT	CTRL-I	20		DC4	CTRL-T
10		LF	CTRL-J	21		NAK	CTRL-U

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
22		SYN	CTRL-V	42		*	*
23		ETB	CTRL-W	43		+	+
24		CAN	CTRL-X	44		,	,
25		EM	CTRL-Y	45		-	-
26		SUB	CTRL-Z	46		.	.
27	[1]	ESC	ESC\ESC	47		/	/
28	[2]	FS	ESC\CTRL--	48		0	0
29	[3]	GS	ESC\CTRL-=	49		1	1
30	[4]	RS	ESC\CTRL-+	50		2	2
31	[5]	US	ESC\CTRL-*	51		3	3
32		Leerzeichen	Leertaste	52		4	4
33		!	SHIFT-1	53		5	5
34		"	SHIFT-2	54		6	6
35		#	SHIFT-3	55		7	7
36		$	SHIFT-4	56		8	8
37		%	SHIFT-5	57		9	9
38		&	SHIFT-6	58		:	SHIFT-;
39		'	SHIFT-7	59		;	;
40		(	SHIFT-9	60		<	<
41		)	SHIFT-0	61		=	=

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
62		>	>	82		R	R
63		?	SHIFT-/	83		S	S
64		@	SHIFT-8	84		T	T
65		A	A	85		U	U
66		B	B	86		V	V
67		C	C	87		W	W
68		D	D	88		X	X
69		E	E	89		Y	Y
70		F	F	90		Z	Z
71		G	G	91		[	SHIFT- ;
72		H	H	92		\	SHIFT- ,
73		I	I	93		]	SHIFT- +
74		J	J	94		↑	SHIFT- *
75		K	K	95		←	SHIFT--
76		L	L	96		'	CTRL-.
77		M	M	97		a	(LOWR) A
78		N	N	98		b	(LOWR) B
79		O	O	99		c	(LOWR) C
80		P	P	100		d	(LOWR) D
81		Q	Q	101		e	(LOWR) E

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
102		f	(LOWR) F	122		z	(LOWR) Z
103		g	(LOWR) G	123		{	CTRL-;
104		h	(LOWR) H	124		|	SHIFT- =
105		i	(LOWR) I	125	[6]	}	ESC\CTRL-< oder ESC\SHIFT-·
106		j	(LOWR) J	126	[7]	~	ESC\BACK S
107		k	(LOWR) K	127	[8]	DEL	ESC\TAB
108		l	(LOWR) L	128			(⅄) CTRL-,
109		m	(LOWR) M	129			(⅄) CTRL-A
110		n	(LOWR) N	130			(⅄) CTRL-B
111		o	(LOWR) O	131			(⅄) CTRL-C
112		p	(LOWR) P	132			(⅄) CTRL-D
113		q	(LOWR) Q	133			(⅄) CTRL-E
114		r	(LOWR) R	134			(⅄) CTRL-F
115		s	(LOWR) S	135			(⅄) CTRL-G
116		t	(LOWR) T	136			(⅄) CTRL-H
117		u	(LOWR) U	137			(⅄) CTRL-I
118		v	(LOWR) V	138			(⅄) CTRL-J
119		w	(LOWR) W	139			(⅄) CTRL-K
120		x	(LOWR) X	140			(⅄) CTRL-L
121		y	(LOWR) Y	141			(⅄) CTRL-M

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
142			(ʌ) CTRL-N	162			(ʌ) SHIFT-2
143			(ʌ) CTRL-O	163			(ʌ) SHIFT-3
144			(ʌ) CTRL-P	164			(ʌ) SHIFT-4
145			(ʌ) CTRL-Q	165			(ʌ) SHIFT-5
146			(ʌ) CTRL-R	166			(ʌ) SHIFT-6
147			(ʌ) CTRL-S	167			(ʌ) SHIFT-7
148			(ʌ) CTRL-T	168			(ʌ) SHIFT-9
149			(ʌ) CTRL-U	169			(ʌ) SHIFT-0
150			(ʌ) CTRL-V	170			(ʌ) *
151			(ʌ) CTRL-W	171			(ʌ) +
152			(ʌ) CTRL-X	172			(ʌ) ,
153			(ʌ) CTRL-Y	173			(ʌ) -
154			(ʌ) CTRL-Z	174			(ʌ) .
155	EOL[9]		(ʌ) RETURN	175			(ʌ) /
156	[10]		ESC\SHIFT-BACK S	176			(ʌ) 0
157	[11]		ESC\SHIFT->	177			(ʌ) 1
158	[12]		ESC\CTRL-TAB	178			(ʌ) 2
159	[13]		ESC\SHIFT-TAB	179			(ʌ) 3
160			(ʌ) Leertaste	180			(ʌ) 4
161			(ʌ) SHIFT-1	181			(ʌ) 5

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
182			(ʌ) 6	202			(ʌ) J
183			(ʌ) 7	203			(ʌ) K
184			(ʌ) 8	204			(ʌ) L
185			(ʌ) 9	205			(ʌ) M
186			(ʌ) SHIFT-;	206			(ʌ) N
187			(ʌ) ;	207			(ʌ) O
188			(ʌ) <	208			(ʌ) P
189			(ʌ) =	209			(ʌ) Q
190			(ʌ) >	210			(ʌ) R
191			(ʌ) SHIFT-/	211			(ʌ) S
192			(ʌ) SHIFT-8	212			(ʌ) T
193			(ʌ) A	213			(ʌ) U
194			(ʌ) B	214			(ʌ) V
195			(ʌ) C	215			(ʌ) W
196			(ʌ) D	216			(ʌ) X
197			(ʌ) E	217			(ʌ) Y
198			(ʌ) F	218			(ʌ) Z
199			(ʌ) G	219			(ʌ) SHIFT-,
200			(ʌ) H	220			(ʌ) SHIFT-+
201			(ʌ) I	221			(ʌ) SHIFT-.

Tabelle D-1: ATASCII-Kodes, ATASCII-Zeichen und die Funktion der Tastatur (Forts.)

Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)	Dezimal-kode	ATASCII-Zeichen	ASCII-Zeichen (falls vorhanden)	Taste(n), die zur Erzeugung dieses Zeichens gedrückt werden muß (müssen)
222			(⇑) SHIFT-*	239			(⇑) (LOWR) O
223			(⇑) SHIFT--	240			(⇑) (LOWR) P
224			(⇑) CTRL-.	241			(⇑) (LOWR) Q
225			(⇑) (LOWR) A	242			(⇑) (LOWR) R
226			(⇑) (LOWR) B	243			(⇑) (LOWR) S
227			(⇑) (LOWR) C	244			(⇑) (LOWR) T
228			(⇑) (LOWR) D	245			(⇑) (LOWR) U
229			(⇑) (LOWR) E	246			(⇑) (LOWR) V
230			(⇑) (LOWR) F	247			(⇑) (LOWR) W
231			(⇑) (LOWR) G	248			(⇑) (LOWR) X
232			(⇑) (LOWR) H	249			(⇑) (LOWR) Y
233			(⇑) (LOWR) I	250			(⇑) (LOWR) Z
234			(⇑) (LOWR) J	251			(⇑) CTRL-;
235			(⇑) (LOWR) K	252			(⇑) SHIFT-=
236			(⇑) (LOWR) L	253	[14]		ESC\CTRL-2
237			(⇑) (LOWR) M	254	[15]		(⇑) ESC\ CTRL-BACK S
238			(⇑) (LOWR) N	255	[16]		(⇑) ESC\CTRL->

Anmerkungen:

[1] [ESC] steht für ein Controlzeichen, daß normalerweise keine Funktion hat; PRINT CHR$(27) ergibt keine sichtbare Veränderung auf dem Bildschirm. Wenn jedoch das nächste Zeichen erneut ein Controlzeichen ist (ATASCII-Kodes 27, 28, 29, 30, 31, 125, 126, 127, 156, 157, 158, 159, 253, 254 oder 255), dann wird nicht die entsprechende Controlfunktion ausgeführt, sondern stattdessen das Zeichen auf dem Bildschirm dargestellt, das dieses Controlzeichen repräsentiert.

[2] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor um eine Zeile nach oben bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[3] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor um eine Zeile nach unten bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[4] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor um eine Spalte nach links bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[5] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor um eine Spalte nach rechts bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[6] repräsentiert das Controlzeichen, mit dessen Hilfe der Bildschirm gelöscht und der Cursor an die obere linke Ecke des Bildschirmes bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall wird der Bildschirm nicht gelöscht.

[7] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor um eine Spalte nach links bewegt und das dort befindliche Zeichen durch ein Leerzeichen ersetzt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[8] repräsentiert das Controlzeichen, mit dessen Hilfe der Cursor an den nächsten Tabulator-Stop bewegt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. Die Position des Cursors verändert sich in diesem Fall nicht.

[9] Hierbei handelt es sich um das ATASCII-EOL-Zeichen.

[10] repräsentiert das Controlzeichen, mit dessen Hilfe die Zeile, in der sich der Cursor gerade befindet, gelöscht wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall wird keine Zeile gelöscht.

[11] repräsentiert das Controlzeichen, mit dessen Hilfe unterhalb der Zeile, in der sich der Cursor gerade befindet, eine neue Zeile eingefügt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall wird keine Zeile eingefügt.

[12] repräsentiert das Controlzeichen, mit dessen Hilfe der Tabulator-Stop an der Stelle, an der sich der Cursor gerade befindet, gelöscht wird (falls vorhanden). Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall werden die Tabulator-Stops nicht beeinflußt.

[13] repräsentiert das Controlzeichen, mit dessen Hilfe an der momentanen Cursorposition ein Tabulator-Stop eingefügt wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall wird kein Tabulator-Stop eingefügt.

[14] repräsentiert das Controlzeichen, mit dessen Hilfe der eingebaute Lautsprecher angesprochen wird. Auf dem Bildschirm geschieht nichts. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein auf dem Bildschirm dargestellt. In diesem Fall wird der eingebaute Lautsprecher nicht angesprochen.

[15] ◀ repräsentiert das Controlzeichen, mit dessen Hilfe das rechts vom Cursor stehende Zeichen gelöscht und der Rest der logischen Zeile um eine Stelle nach links verschoben wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein ◀ auf dem Bildschirm dargestellt. In diesem Fall wird kein Zeichen gelöscht.

[16] ▶ repräsentiert das Controlzeichen, mit dessen Hilfe rechts vom Cursor ein Leerzeichen eingefügt und der Rest der logischen Zeile um ein Zeichen nach rechts verschoben wird. Wenn dieses Controlzeichen an den Bildschirm übertragen wird und unmittelbar vorher ein Zeichen mit dem ATASCII-Kode 27 ausgegeben wurde, dann wird nicht die Controlfunktion ausgeführt, sondern ein ▶ auf dem Bildschirm dargestellt. In diesem Fall wird kein Zeichen eingefügt.

Anhang E
DIE BASIC-BEFEHLE UND -FUNKTIONEN UND DIE DAZUGEHÖRIGEN ABKÜRZUNGEN

Bef. od. Funkt.	Abkürzung	Bef. od. Funkt.	Abkürzung	Bef. od. Funkt.	Abkürzung
ABS		GOTO	G.	PUT	PU.
ADR		GRAPHICS	GR.	RAD	
AND		IF		READ	REA.
ASC		INPUT	I.	REM	R. or .
ATN		INT		RESTORE	RES.
BYE	B.	LEN		RETURN	RET.
CLOAD	CLOA.	LET	LE.	RND	
CHR$		LIST	L.	RUN	RU.
CLOG		LOAD	LO.	SAVE	S.
CLOSE	CL.	LOCATE	LOC.	SETCOLOR	SE.
CLR		LOG		SGN	
COLOR	C.	LPRINT	LP.	SIN	
COM		NEW		SOUND	SO.
CONT	CON.	NEXT	N.	SQR	
COS		NOT		STATUS	ST.
CSAVE	CS.	NOTE	NO.	STEP	
DATA	D.	ON		STICK	
DEG	DE.	OPEN	O.	STRIG	
DIM	DI.	OR		STOP	STO.
DOS	DO.	PADDLE		STR$	
DRAWTO	DR.	PEEK		THEN	
END		PLOT	PL.	TO	
ENTER	E.	POINT	P.	TRAP	T.
EXP		POKE	POK.	USR	
FOR	F.	POP		VAL	
FRE		POSITION	POS.	XIO	X.
GET	GE.	PRINT	PR. or ?		
GOSUB	GOS.	PTRIG			

Anhang F
DIE SPEICHERAUFTEILUNG

Der Speicher des ATARI-Computers kann in drei Teile unterteilt werden: den Schreib-Lese-Speicher (Random Access Memory = RAM), den Nur-Lese-Speicher (Read-Only Memory = ROM) und einen Speicherbereich, der zur Ein- und Ausgabe verwendet wird (Input/Output = I/O). Bild F-1 können Sie entnehmen, wie der Speicher beim ATARI-Computer aufgeteilt ist. In den anderen Bildern und Tabellen, die in diesem Abschnitt enthalten sind, werden Teile dieses Bildes etwas detaillierter dargestellt.

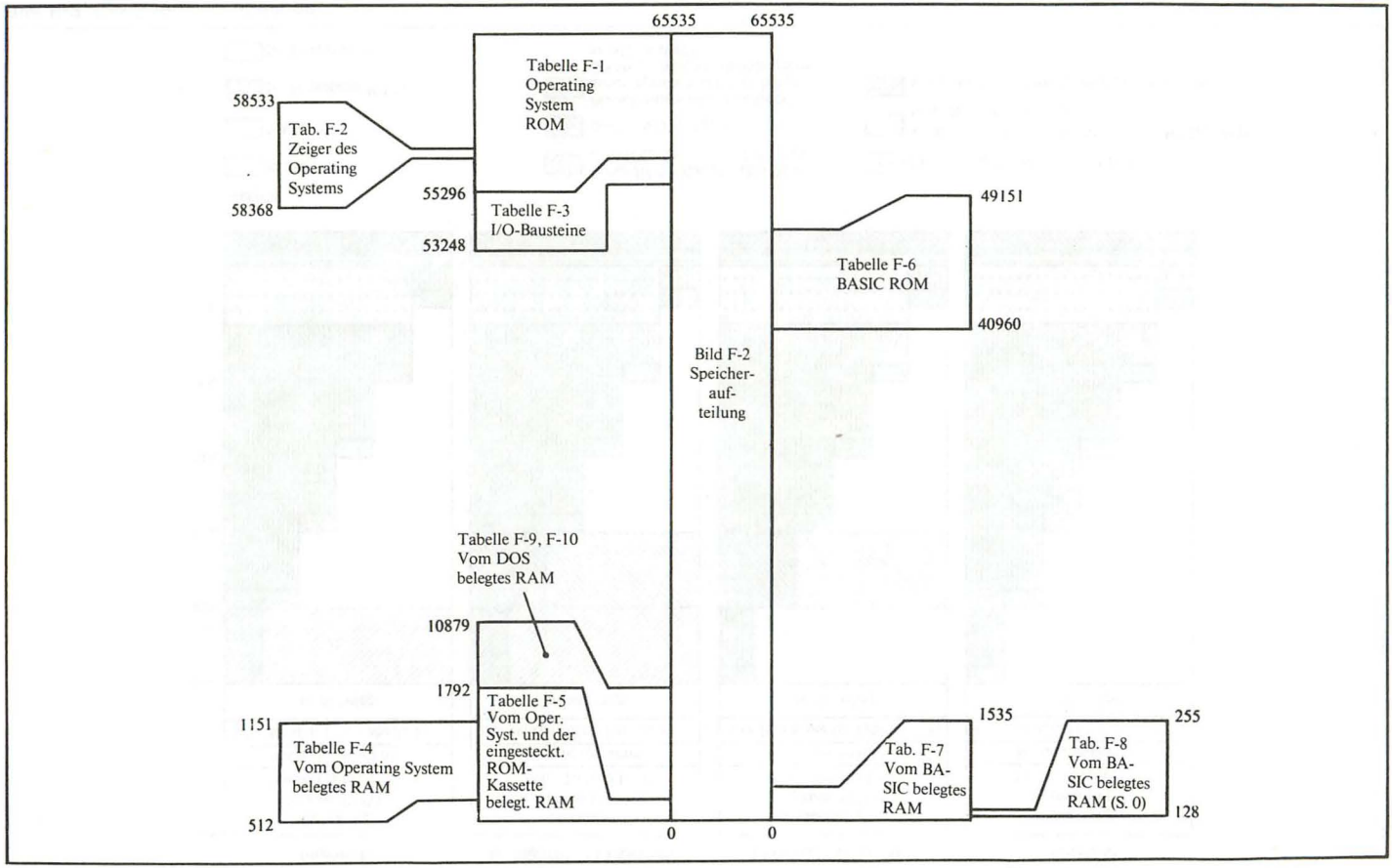


Bild F-1: Die Speicherbelegung beim ATARI 400/800

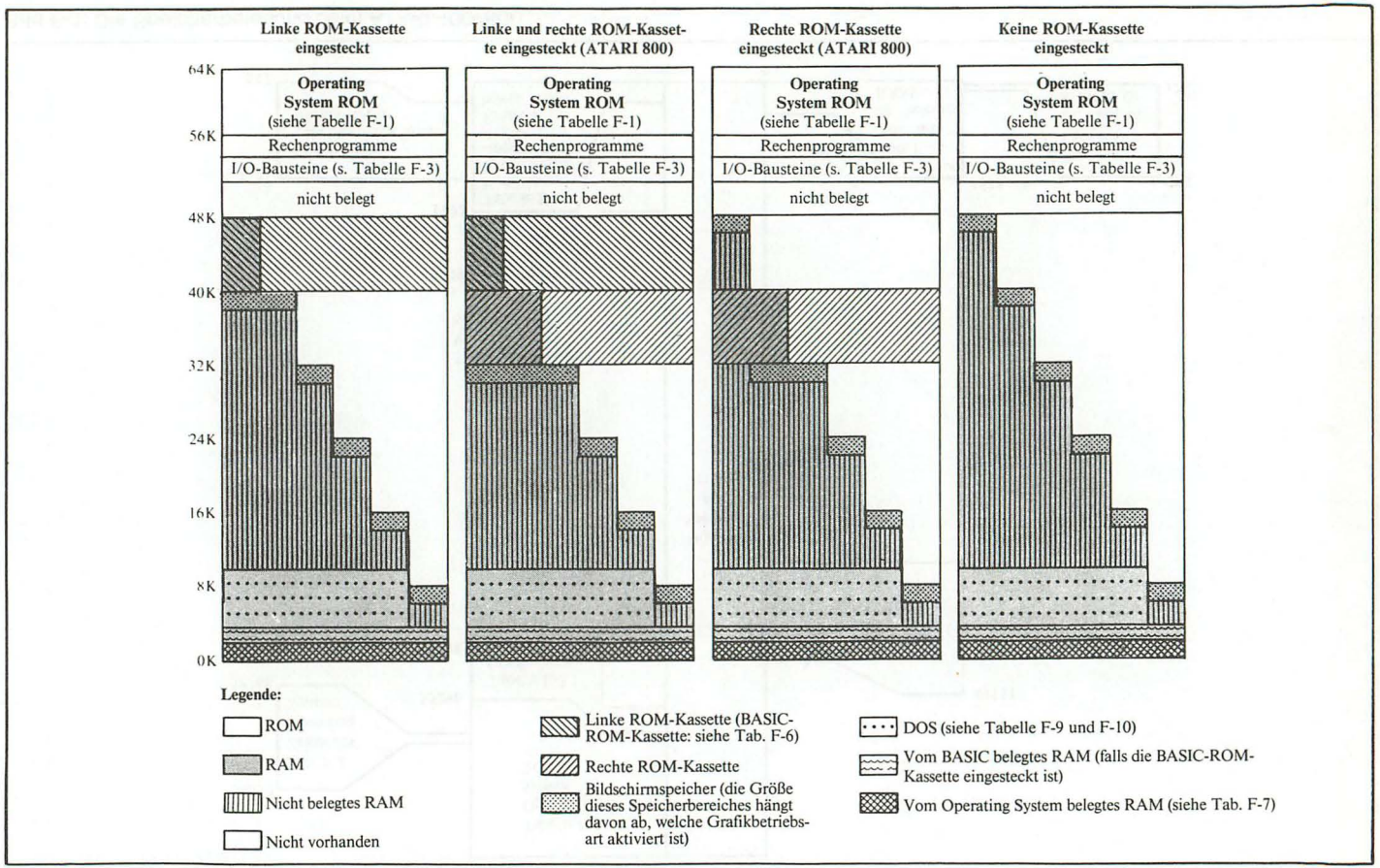


Bild F-2: Die Speicheraufteilung

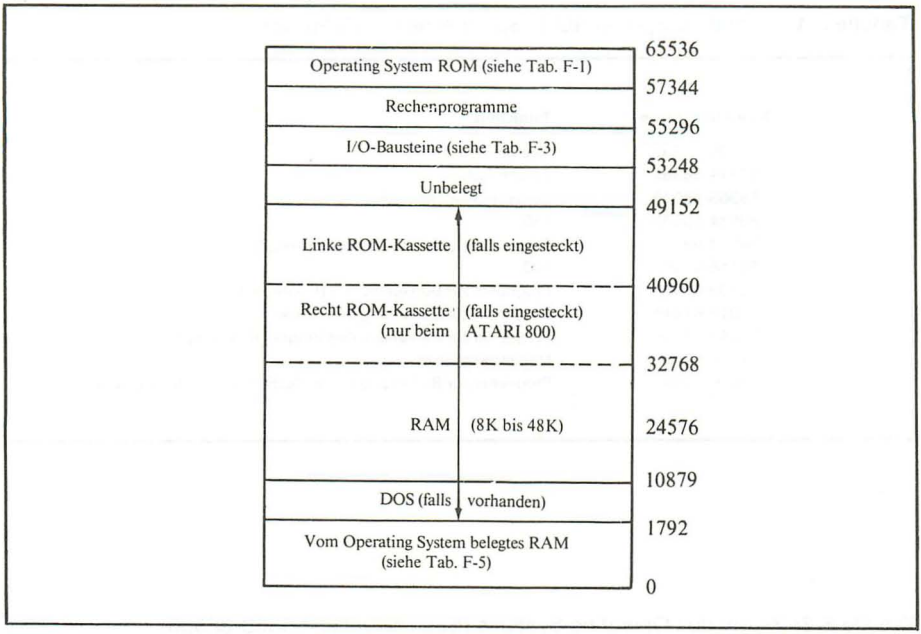


Bild F-3: Speicherbelegung, wenn die BASIC-ROM-Kassette nicht eingesteckt ist.

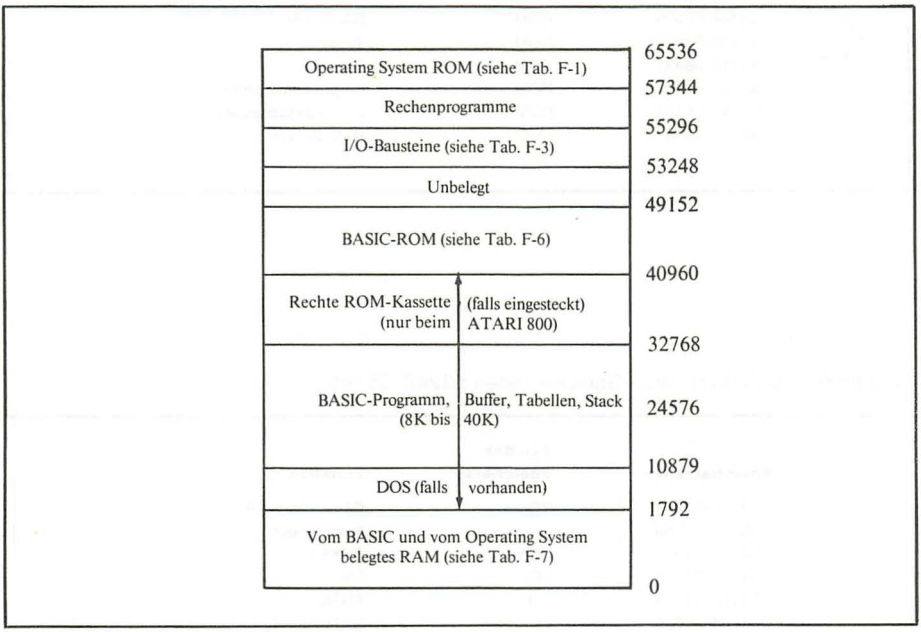


Bild F-4: Speicherbelegung, wenn die BASIC-ROM-Kassette eingesteckt ist

Tabelle F-1: Operating System ROM (Speicherzellen 55296-65535)

Speicherzellen	Funktion
55296-57393	Rechenprogramme
57344-58367	Zeichensatz
58368-58533	Zeiger (siehe Tabelle F-2)
58534-59092	CIO
59093-59715	Programm zur Bedienung des Interrupts
59716-60905	SIO
60906-61047	Programm zur Bedienung der Diskettenstation
61048-61248	Programm zur Bedienung des Druckers
61249-61666	Programm zur Bedienung des Programmrecorders
61667-62435	Monitorprogramm
62436-65535	Programm zur Bedienung des Bildschirmes und der Tastatur

Tabelle F-2: Zeiger des Operating Systems (Speicherzellen 58368-68533)

Speicherzellen	Typ des Speichers	Funktion
58368-58383	ROM	Editor
58384-58399	ROM	Bildschirm
58400-58415	ROM	Tastatur
58416-58431	ROM	Drucker
58432-58447	ROM	Programmrecorder
58448-58495	ROM	Einsprungadressen
58496-58533	ROM	RAM-Zeiger

Tabelle F-3: I/O-Bausteine (Speicherzellen 53248-55295)

Speicherzellen	Typ des Speichers	Funktion
53248-53503	I/O	CTIA oder GTIA
53504-53759	I/O	Nicht belegt
54760-54015	I/O	POKEY
54016-54271	I/O	PIA
54272-54783	I/O	ANTIC
54784-55295	I/O	Nicht belegt

Tabelle F-4: Vom Operating System belegtes RAM (Speicherzellen 512-1151)

Speicherzellen	Funktion
512-553	Interrupt-Zeiger
554-623	Sonstiges
624-647	Spielregler
648-655	Sonstiges
656-703	Bildschirmspeicher (je nachdem, welche Grafikbetriebsart aktiviert ist)
704-711	Farben
712-735	Keine Funktion
736-767	Sonstiges
768-779	DCB
780-793	Sonstiges
794-831	Adressen der Programme zur Bedienung der externen Geräte
832-847	Ein/Ausgabekanal 0 (IOCB0)
848-863	Ein/Ausgabekanal 1 (IOCB1)
864-879	Ein/Ausgabekanal 2 (IOCB2)
880-895	Ein/Ausgabekanal 3 (IOCB3)
896-911	Ein/Ausgabekanal 4 (IOCB4)
912-927	Ein/Ausgabekanal 5 (IOCB5)
928-943	Ein/Ausgabekanal 6 (IOCB6)
944-959	Ein/Ausgabekanal 7 (IOCB7)
960-999	Druckerbuffer
1000-1020	Keine Funktion
1021-1151	Kassettenbuffer

Tabelle F-5: Vom Operating System oder der eingesteckten ROM-Kassette belegtes RAM (Speicherzellen 0-2047)

Speicherzellen	Funktion
0-127	Vom Operating System belegtes RAM (Seite 0)
128-255	Für den Anwender verfügbares RAM (Seite 0)
256-511	Stack
512-1151	Vom Operat. Syst. belegtes RAM (siehe Tab. F-4)
1152-1791	Für den Anwender verfügbares RAM
1792-2047	Für den Anwender verfügbares BOOT-RAM

Tabelle F-6: BASIC-ROM (Speicherzellen 40960-49151)*

Speicherzellen	Funktion
40960-41036	Kaltstart
41037-41055	Warmstart
41056-42081	Syntax
42082-42158	Absuchen
42159-42508	Tabelle für die Befehlsworte
42509-43134	Syntaxtabelle
43135-43358	Speicherverwaltung
43359-43519	Ausführung des CONT-Befehls
43520-43631	Befehlstabelle
43632-43743	Tabelle für die Operationszeichen
43744-44094	Berechnen eines Ausdrucks
44095-44163	Hirarchie der Operationszeichen
44164-45001	Verarbeitung der Operationszeichen
45002-45320	Berechnen einer Funktion
45321-47127	Ausführen eines Befehls
47128-47381	Unterprogramme für den CONT-Befehl
47382-47542	Fehler
47543-47732	Grafik
47733-48548	I/O-Programme
48549-49151	Rechenprogramme

* Gilt nur für das normale ATARI-BASIC.

Tabelle F-7: Vom BASIC belegtes RAM (Speicherzellen 0-255F)*

Speicherzellen	Funktion
0-127	Vom Operating System belegtes RAM (Seite 0)
128-255	Vom BASIC belegtes RAM (Seite 0, siehe Tab. F-8)
256-511	Stack
512-1151	Vom Operating System belegtes RAM
1152-1405	Syntax-Stack
1406-1535	Eingabebuffer
1536-1791	Frei
1792-Ende des nicht belegten RAM-Speichers	BASIC-Programm: Syntaxbuffer** Tabelle mit Variablennamen** Wertetabelle** Programm** Bereich für Felder und alphanumer. Variablen** Stack (wird bei der Programmausführung verwendet)**

* Gilt nur für das normale ATARI-BASIC.

** An welcher Stelle sich dieser Teil befindet, hängt davon ab, wie groß das Programm ist und wieviele Variablen verwendet werden.

Tabelle F-8: Vom BASIC belegtes RAM (Seite 0; Speicherzellen 128-255)

Speicherzellen	Funktion
128-145	Zeiger des Programms
146-202	Sonstiges
203-209	Nicht belegt
210-255	Arbeitsbereich der Rechenprogramme

Tabelle F-9: Von DOS 1.0 und der Fileverwaltung (File Management System = FMS) belegtes RAM

Speicherzellen	Funktion
1792-4863	Vom File Management System belegtes RAM
4864-9855	Vom Disk Operating System belegtes RAM
9856-10879	Diskettenbuffer

Tabelle F-10: Von DOS 2.0S und der Fileverwaltung (File Management System = FMS) belegtes RAM

Speicherzellen	Funktion
1792-4863	Vom File Management System belegtes RAM
4864-9855	Vom Disk Operating System belegtes RAM
9856-10879	Laufwerksbuffer 1-4 und Sektorbuffer 1-2
10880-LOMEM	Disk Operating System Hilfsprogramme (Sektorbuffer 3-7)

Anhang G

EINIGE OFT BENUTZTE SPEICHERADRESSEN UND DEREN FUNKTIONEN

Viele Speicherzellen erfüllen ganz bestimmte Aufgaben. In diesem Abschnitt sind die Speicherzellen aufgeführt, die für den BASIC-Programmierer von besonderem Interesse sind. Auf die Speicherzellen, die hier nicht enthalten sind, wird entweder bei der Ausführung eines bestimmten BASIC-Befehls automatisch zugegriffen oder sie erfüllen keine besonders interessante Funktion. Mit Hilfe eines PEEK-Befehls kann der Inhalt einer Speicherzelle gelesen und mit Hilfe eines POKE-Befehls der Inhalt einer Speicherzelle verändert werden.
In BASIC werden Speicheradressen und Inhalte von Speicherzellen dezimal angegeben. Die Speicherzellen sind von 0 bis 65535 numeriert. Der Wert des Inhaltes einer Speicherzelle kann zwischen 0 und 255 liegen. Wenn größere Werte gespeichert werden sollen, benötigt man zwei aufeinanderfolgende Speicherzellen. Der Inhalt dieser beiden Speicherzellen zusammen berechnet sich aus der Summe des Inhaltes der ersten Speicherzelle und dem 256-fachen des Inhaltes der zweiten Speicherzelle. Um die Spalte zu bestimmen, an der sich der Cursor im Moment befindet, muß zum Beispiel PEEK(85) + 256 * PEEK(86) berechnet werden. Umgekehrt müssen, um die Spaltenposition des Cursors zu verändern, die beiden Befehle POKE 85, SPAL-INT (SPAL/256) * 256 und POKE 86, INT (SPAL/256) * 256 ausgeführt werden. Anschließend befindet sich der Cursor in der Spalte SPAL.
Für einige Speicherzellen wird häufig auch ein bestimmter Name verwendet. Diese Namen sind in Klammern unter der Bezeichnung der entsprechenden Speicherzelle angegeben.

Speicheraufteilung

14,15 Unteres Ende der Display List und des Bildschirmspeichers (Display Screen Lower Limit = APPMHI)

Diese Speicherzellen enthalten die höchste Speicheradresse, die noch zur Speicherung eines BASIC-Programmes oder von Variablen verwendet werden kann. Alles, was darüber liegt, ist für die Display List und den Bildschirmspeicher reserviert.

88,89 Adresse des Bildschirmspeichers
(Screen Memory Address = SAVMSC)

Diese Speicherzelle enthält die unterste Adresse des Bildschirmspeichers. Der Inhalt dieser Adresse wird in der oberen linken Ecke des Bildschirmes dargestellt.

106 Oberes Ende des RAM-Speichers (höherwertiges Byte)
(Top of RAM Address = RAMTOP)

Der Inhalt dieser Speicherzelle entspricht der 16-fachen Anzahl der 4K RAM-Blöcke, auf die der Computer ausgebaut ist. Die Anzahl der 1K-Blöcke kann mit PEEK (740)/4 berechnet werden.

741,742 Oberes Ende des freien Speichers
(Free Memory High Address = MEMTOP)

Die höchste Adresse des freien Speichers kann mit PEEK(741) + 256 * PEEK(742) – 1 berechnet werden. Der Inhalt dieser Speicherzellen kann sich verändern, wenn die SYSTEM RESET-Taste gedrückt oder ein Ein/Ausgabekanal für den Bildschirm geöffnet wird.

743,744 Unteres Ende des freien Speichers
(Free Memory Low Address = MEMLO)

Diese Speicherzellen enthalten die niedrigste Adresse des freien Speichers. Der Inhalt dieser Speicherzellen kann sich beim Drücken der SYSTEM RESET-Taste verändern.

Bildschirmverwaltung

77 Farbwechsel Ein/Aus
(Attract Mode On/Off = ATRACT)

Wenn in dieser Speicherzelle eine 0 abgelegt wird, dann wird der Farbwechsel auf dem Bildschirm ausgeschaltet. Das geschieht automatisch immer dann, wenn eine Taste der Tastatur gedrückt wird. Wird in dieser Speicherzelle jedoch eine 254 abgelegt, so wird der Farbwechsel eingeschaltet. Das geschieht automatisch, wenn neun Minuten lang keine Taste gedrückt wurde.

82 Linke Bildschirmbegrenzung
(Left Margin of Text Area = LMARGN)

Diese Speicherzelle enthält die Nummer der Spalte, an der sich in der Grafikbetriebsart 0 die linke Bildschirmbegrenzung befindet. PEEK(82) liegt immer zwischen 0 und 39. 0 entspricht dem linken Bildschirmrand. Normalerweise ist in dieser Speicherzelle eine 2 abgelegt.

83 Rechte Bildschirmbegrenzung
(Right Margin of Text Area = RMARGN)

Diese Speicherzelle enthält die Nummer der Spalte, an der sich in der Grafikbetriebsart 0 die rechte Bildschirmbegrenzung befindet. PEEK (83) liegt immer zwischen 0 und 39. 39 entspricht dem rechten Bildschirmrand. Normalerweise ist in dieser Speicherzelle eine 39 abgelegt.

84 Momentane Cursorposition (Zeile)
(Current Row Cursor Position = ROWCRS)

Der Inhalt dieser Speicherzelle entspricht der Zeile des Bildschirmes, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. Der Minimalwert von PEEK(84) ist 0. Wie groß der Maximalwert ist, hängt davon ab, welche Grafikbetriebsart aktiviert ist (siehe Tabelle 11-7).

85,86 Momentane Cursorposition (Spalte)
(Current Column Cursor Position = COLCRS)

Der Inhalt dieser Speicherzellen entspricht der Spalte des Bildschirmes, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. Der Minimalwert von PEEK (85) ist 0. Wie groß der Maximalwert ist, hängt davon ab, welche Grafikbetriebsart aktiviert ist (siehe Tabelle 11-7). In den Grafikbetriebsarten 0 bis 7 ist PEEK(86) immer 0.

87 Grafikbetriebsart
(Display Mode = DINDEX)

Mit Hilfe dieser Speicherzelle wird festgelegt, welche Grafikbetriebsart gerade auf dem Bildschirm dargestellt wird.

90 Grafik-Anfangspunkt (Zeile)
(Starting Graphics Cursor Row = OLDROW)

Diese Speicherzelle enthält die Nummer der Bildschirmzeile, an der mit der Ausführung eines DRAWTO- oder XIO 18-Befehls begonnen wird.

91,92 Grafik-Anfangspunkt (Spalte)
(Starting Graphics Cursor Column = OLDCOL)

Diese Speicherzellen enthalten die Nummer der Spalte des Bildschirmes, an der mit der Ausführung eines DRAWTO- oder XIO 18-Befehls begonnen wird.

93 Zeichen unter Cursor
(Cursor Character Save/Restore = OLDCHR)

In dieser Speicherzelle ist das Zeichen gespeichert, das durch den Cursor verdeckt ist. Mit ihrer Hilfe wird das Zeichen wieder auf den Bildschim gebracht, nachdem sich der Cursor an eine andere Stelle bewegt hat.

94,95 Cursoradresse
(Cursor Memory Address = OLDADR)

In dieser Speicherzelle ist die Adresse des Cursors gespeichert. Zusammen mit OLDCHR wird diese Speicherzelle dazu benutzt, das durch den Cursor verdeckte Zeichen wieder auf den Bildschirm zu bringen, nachdem sich der Cursor an eine andere Stelle bewegt hat.

96 Grafik-Endpunkt (Zeile)
(Ending Graphics Cursor Row = NEWROW)

Diese Speicherzelle enthält die Nummer der Bildschirmzeile, an der die Ausführung eines DRAWTO- oder XIO 18-Befehls endet.

97,98 Grafik-Endpunkt (Spalte)
(Ending Graphics Cursor Column = NEWCOL)

Diese Speicherzellen enthalten die Nummer der Spalte des Bildschirmes, an der die Ausführung eines DRAWTO- oder XIO 18-Befehls endet.

201 Abstand zwischen Tabulator-Stops (bei Ausgabe auf den Bildschirm)
(Display Screen Tab Interval = PTABW)

Mit Hilfe dieser Speicherzelle wird festgelegt, wie groß der Abstand (in Spalten) zwischen zwei Tabulator-Stops sein soll. Der erste Tabulator-Stop befindet sich an der Spalte, deren Nummer dem Inhalt dieser Speicherzelle entspricht. Normalerweise ist hier eine 10 abgelegt.

656 Cursorposition im Textfenster (Zeile)
(Text Cursor Row Position = TXTROW)

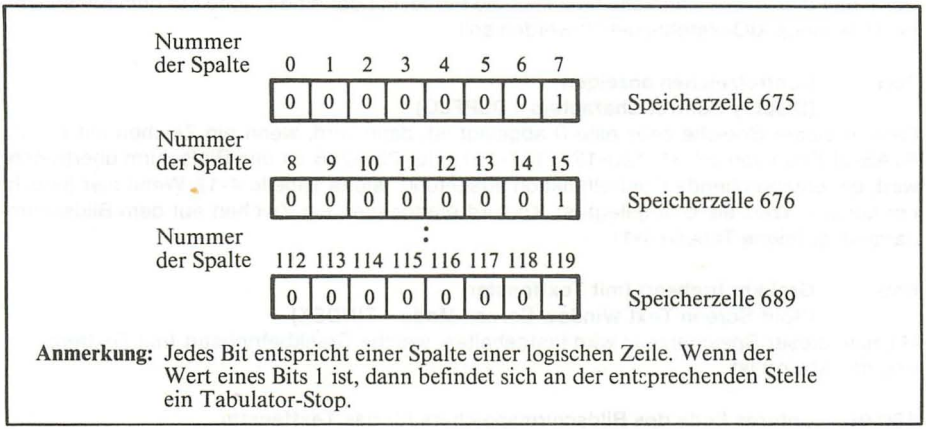
Der Inhalt dieser Speicherzelle entspricht der Zeile des Textfensters, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. PEEK(656) liegt immer zwischen 0 und 3. 0 entspricht der ersten Zeile des Textfensters.

657,658 Cursorposition im Textfenster (Spalte)
(Text Cursor Column Position = TXTCOL)

Der Inhalt dieser Speicherzellen entspricht der Spalte des Textfensters, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. PEEK(657) liegt immer zwischen 0 und 39. 0 entspricht der ersten Spalte des Textfensters. Sofern Sie den Inhalt der Speicherzelle 658 nicht ändern, ist dort immer eine 0 abgelegt.

675-689 Tabelle zur Festlegung der Tabulator-Stops
(Display Screen Tab Stop Map = TABMAP)

Mit Hilfe von 15 Bytes (120 Bits) wird festgelegt, an welchen Stellen einer logischen Zeile sich Tabulator-Stops befinden sollen. Jedes Bit entspricht einer Spalte einer logischen Zeile. Wenn der Wert eines Bits 1 ist, dann befindet sich an der entsprechenden Stelle ein Tabulator-Stop (siehe Bild G-1). Jedesmal, wenn ein Ein/Ausgabekanal für den Bildschirm geöffnet wird (Gerätename S: oder E:), wird in jedem Byte dieser Tabelle eine 1 abgelegt, was zur Folge hat, daß sich an den Spalten 7, 15, 23 usw. Tabulator-Stops befinden.



Anmerkung: Jedes Bit entspricht einer Spalte einer logischen Zeile. Wenn der Wert eines Bits 1 ist, dann befindet sich an der entsprechenden Stelle ein Tabulator-Stop.

Bild G-1: Tabelle zur Festlegung der Tabulator-Stops

Tabelle G-1: Die verschiedenen Möglichkeiten der Cursor- und Zeichendarstellung

Inhalt von Speicherzelle 755 (dezimal)	Cursor				Zeichendarstellung	
	Durchsichtig	Undurchsichtig	Sichtbar	Unsichtbar	Normal	Auf dem Kopf stehend
0	X			X	X	
1		X		X	X	
2	X		X		X	
3		X	X		X	
4	X			X		X
5		X		X		X
6	X		X			X
7		X	X			X

752 Cursorunterdrückung
(Cursor Inhibit = CRSINH)

Wenn in dieser Speicherzelle eine 1 abgelegt ist, dann ist der Cursor sichtbar. Jeder andere Wert als 0 macht den Cursor unsichtbar.

755 Festlegung der Cursor- und Zeichendarstellung
(Character and Cursor Control = CHACT)

In dieser Speicherzelle ist normalerweise eine 2 abgelegt. Durch Verändern des Inhaltes dieser Speicherzelle kann der Cursor undurchsichtig oder unsichtbar gemacht oder die Zeichendarstellung umgekehrt werden. In Tabelle G-1 sind alle Möglichkeiten aufgeführt.

756 Auswahl des Zeichensatzes
(Character Adress Base = CHBAS)

Mit Hilfe dieser Speicherzelle wird ausgewählt, welcher Zeichensatz in den Grafikbetriebsarten 1 und 2 benutzt werden soll. Wenn hier eine 224 abgelegt ist, dann wird der Zeichensatz benutzt, in dem die Großbuchstaben und Zahlen enthalten sind. Wenn jedoch in dieser Speicherzelle eine 226 gespeichert ist, so wird der Zeichensatz benutzt, in dem die Kleinbuchstaben und Grafikzeichen enthalten sind.

765 Auswahl der Zeichenfarbe für den XIO-Befehl
(Fill Data = FILDAT)

Der Inhalt dieser Speicherzelle bestimmt die Farbe, mit der eine Fläche auf dem Bildschirm mit Hilfe eines XIO-Befehls gefüllt werden soll.

766 Controlzeichen anzeigen
(Display Control Characters = DSPFLG)

Falls in dieser Speicherzelle eine 0 abgelegt ist, dann wird, wenn ein Zeichen mit einem ATASCII-Kode von 27-31, 123-127, 187-191 oder 251-255 an den Bildschirm übertragen wird, die entsprechende Controlfunktion ausgeführt (siehe Tabelle 4-1). Wenn hier jedoch ein anderer Wert als 0 abgelegt ist, so wird stattdessen ein Zeichen auf dem Bildschirm dargestellt (siehe Tabelle 4-1).

659 Grafikbetriebsart (mit Textfenster)
(Split-Screen Text Window Screen Mode = TINDEX)

Mit Hilfe dieser Speicherzelle wird festgehalten, welche Grafikbetriebsart (mit Textfenster) gerade aktiviert ist.

660,661 Unteres Ende des Bildschirmspeichers für das Textfenster
(Split-Screen Memory Address = TXTMSC)

In diesen Speicherzellen ist die niedrigste Adresse des Bildschirmspeichers für das Textfenster festgehalten. Der Inhalt dieser Speicherzelle wird in der oberen linken Ecke des Textfensters dargestellt.

665-667 Zeichen unter Cursor und Cursoradresse (im Textfenster)
(Split-Screen Cursor Data)

Diese Speicherzellen erfüllen im Textfenster dieselbe Funktion, wie sonst OLDCHR (Speicherzelle 93) und OLDADR (Speicherzelle 94 und 95).

763 Latztes ATASCII-Zeichen oder letzter Grafikpunkt
(Last ATASCII Character or Plot Point = ATACHR)

Diese Speicherzelle enthält die ATASCII-Kodezahl des Zeichens, das zuletzt ein- oder ausgegeben wurde beziehungsweise den Wert des Grafikpunktes, der zuletzt gezeichnet wurde. Diese Speicherzelle dient bei der Ausführung eines DRAWTO- oder XIO-Befehls zur Festlegung der Zeichenfarbe.

54273 Cursor- und Zeichendarstellung
(Character Control Register = CHACTL)
Erfüllt dieselbe Funktion wie CHACT (Speicherzelle 755).

Display List

512,513 Display List Interrupt-Zeiger
(Display List Interrupt Vector = VDSLST)

In diesen Speicherzellen ist die Adresse gespeichert, zu der der Computer springt, wenn bei der Ausführung der Display List eine Unterbrechung (Display List Interrupt) auftritt.

559 DMA-Kontrollregister
(DMA Control Register = DMACTL)
Mit Hilfe dieser Speicherzelle wird der direkte Speicherzugriff (DMA) ein- oder ausgeschaltet. Normalerweise ist hier eine 22 abgelegt. In diesem Fall können mit Hilfe von DMA Anweisungen der Display List oder Daten von dem Playfield gelesen werden. Wenn in dieser Speicherzelle eine 0 abgelegt wird, ist kein DMA mehr möglich. In Tabelle 9-3 ist aufgeführt, welche Werte hier gespeichert werden müssen, um Player-Missile-Grafiken zu ermöglichen.

560,561 Adresse der Display List
(Display List Address = SDLNST)
Diese Speicherzellen enthalten die Anfangsadresse der Display List.

54286 Kontrollregister für den nicht maskierbaren Interrupt
(Non-maskable Interrupt Enable = NMIEN)
Mit Hilfe dieser Speicherzelle kann der Display List Interrupt oder der Interrupt, der bei einer Vertikalaustastung des angeschlossenen Fernsehgerätes erzeugt wird, ein- oder ausgeschaltet werden. Wenn hier eine 0 abgelegt wird, kann kein Display List Interrupt mehr erzeugt werden. Bei einer 128 wird bei einer Vertikalaustastung kein Interrupt mehr erzeugt und bei einer 192 ist beides nicht mehr möglich.

Player-Missile-Grafik

623 Prioritätsregister
(Player/Playfield Priorities = GPRIOR)
Mit Hilfe dieser Speicherzelle wird festgelegt, welche Farbe auf dem Bildschirm dargestellt wird, wenn sich ein Player und ein Objekt des Playfields überlappen. Wenn hier eine 1 abgelegt wird, haben alle Player Priorität über das Playfield. Bei einer 2 haben die Player 0 und 1 Priorität über das Playfield sowie über die Player 2 und 3. Bei einer 4 hat das Playfield Priorität über die Player und bei einer 8 haben die Farbregister 0 und 1 des Playfields Priorität über die Player sowie über die Farbregister 2 und 3.

704-707 Player-Missile Farbregister
(Player-Missile Color Registers = COLPMO-COLPM3)
Jede dieser Speicherzellen bestimmt die Farbe eines Players und des entsprechenden Missiles. In Tabelle 9-4 ist aufgeführt, welche Werte hier abgelegt werden können und welche Farben damit erzeugt werden.

53248-53251 Horizontale Position der Player
(Player Horizontal Position Registers = HPOSPO-HPOSP3)
Jede dieser Speicherzellen bestimmt die horizontale Position eines Players. Der Inhalt die-

ser Speicherzellen muß zwischen 0 (linker Bildschirmrand) und 277 (rechter Bildschirmrand) liegen.

53256-53259 Breite der Player
(Player Width Registers = SIZEPO-SIZEP3)

Jede dieser Speicherzellen bestimmt den Vergrößerungsfaktor, der bei der Darstellung eines Players auf dem Bildschirm benutzt werden soll. Wenn hier eine 0 oder eine 2 abgelegt wird, dann wird der Player in normaler Breite dargestellt. Bei einer 1 ist die Breite doppelt und bei einer 3 viermal so groß wie normal.

53277 Grafik-Kontrollregister
(Graphics Control Register = GRACTL)

Zusammen mit DMACTL (Speicherzelle 559) dient diese Speicherzelle zur Aktivierung von DMA für Player-Missile-Grafiken. Wenn hier eine 2 abgelegt wird, wird DMA für die Player aktiviert. Bei einer 1 wird DMA für die Missiles aktiviert und bei einer 3 sowohl für die Player als auch für die Missiles.

54279,54280 Anfangsadresse der Player/Missile-Tabelle
(Player-Missile Base Register = PMBASE)

Diese Speicherzellen enthalten die Anfangadresse der Player/Missile-Tabelle.

Kassettenbuffer

61 Zeiger des Kassettenbuffers
(Cassette Buffer Pointer = BPTR)

Diese Speicherzelle enthält einen Zeiger zur Adressierung der Stelle des Kassettenbuffers, auf die als nächstes zugegriffen wird. Der Inhalt dieser Speicherzelle kann zwischen 0 und dem Wert, der in BLIM (Speicherzelle 650) abgelegt ist, liegen. Wenn BPTR = BLIM ist, dann ist der Kassettenbuffer voll, falls Daten ausgegeben wurden oder leer, wenn Daten eingelesen wurden.

63 End-of-File von Kassette gelesen
(Cassette End-of-File Flag = FEOF)

Diese Speicherzelle wird dazu benutzt, um festzuhalten, ob ein EOF-Zeichen von der Kassette gelesen wurde. Wenn dem so ist, dann befindet sich in dieser Speicherzelle irgendeine Zahl (jedoch nicht 0). Falls bisher noch kein EOF-Zeichen gelesen wurde, dann ist hier eine 0 abgelegt.

64 Kontrolltonzähler
(Beep Count = FREQ)

Mit Hilfe dieser Speicherzelle wird festgehalten, wieviele Kontrolltöne der Computer erwartet.

649 Daten auf Kassette schreiben oder von Kassette lesen
(Cassette Read/Write Mode Flag = WMODE)

Wenn in dieser Speicherzelle eine abgelegt ist, dann werden Daten von der Kassette gelesen. Bei einer 128 werden Daten auf Kassette geschrieben.

650 Größe des Kassettenbuffers
(Cassette Buffer Size = BLIM)

Diese Speicherzelle enthält die Anzahl der Bytes, die sich momentan im Kassettenbuffer befinden. Der Wert von BLIM kann zwischen 0 und 128 liegen.

1021-1151 Kassettenbuffer
(Cassette Buffer = CASBUF)
Diese Speicherzellen benutzt der Computer beim Lesen oder Schreiben von beziehungsweise auf Kassette als Buffer.

Tastatur

17 BREAK-Taste gedrückt
(BREAK Key Flag = BRKKEY)
Wenn sich in dieser Speicherzelle eine 0 befindet, wurde die BREAK-Taste gedrückt.

694 Inverse Zeichendarstellung
(Inverse Video Keystrokes = INVFLG)
Wenn sich in dieser Speicherzelle eine 0 befindet, dann wird, wenn eine Taste der Tastatur gedrückt wird, der ATASCII-Kode eines normal dargestellten Zeichens erzeugt. Falls hier jedoch ein anderer Wert als 0 gespeichert ist, so wird stattdessen der ATASCII-Kode eines invers dargestellten Zeichens erzeugt.

702 Umschaltung Groß/Kleinschreibung oder Controlzeichen
(Shift/Control Lock Flag = SHFLOK)
Sinnvolle Inhalte dieser Speicherzelle sind 0 (normale Groß/Kleinschreibung), 64 (Großschreibung) oder 128 (Controlzeichen).

764 Zuletzt gedrückte Taste
(Keyboard Character = CH)
In dieser Speicherzelle ist die Kodezahl der Taste gespeichert, die als letzte gedrückt wurde. Falls keine Taste gedrückt wurde, ist hier eine 255 abgelegt.

767 Bildschirmausgabe gestoppt
(Start/Stop Display Screen = SSFLAG)
Wenn in dieser Speicherzelle eine 255 abgelegt ist, dann ist die Bildschirmausgabe gestoppt. Falls hier eine 0 gespeicher ist, wird die Bildschirmausgabe nicht beeinflußt. Beim Drücken von CTRL-1 wird der Inhalt dieser Speicherzelle verändert.

53279 Betätigung der Funktionstasten
(CONSOLE Switch Port = CONSOL)
Diese Speicherzelle erfüllt zwei Aufgaben. Erstens kann mit Hilfe eines PEEK(53279)-Befehls festgestellt werden, ob und wenn ja, welche der drei Funktionstasten (OPTION, SELECT und START) gedrückt ist. Vor einem PEEK(53279)-Befehl sollte allerdings der Befehl POKE 53279,8 ausgeführt werden. Nur dann ist vollkommen sicher, daß der gelesene Wert korrekt ist. In Tabelle G-2 ist aufgeführt, welche Werte gelesen werden, wenn eine oder mehrere bestimmte Funktionstasten gedrückt sind.
Die zweite Funktion dieser Speicherzelle besteht darin, daß, wenn die beiden Befehle POKE 53279,0 und POKE 53279,8 nacheinander ausgeführt werden, der eingebaute Lautsprecher angesprochen wird.

Tonerzeugung

65 Tonerzeugung bei Ein/Ausgabe
(Input/Output Noise Control = SOUNDR)
Normalerweise ist in dieser Speicherzelle irgendeine Zahl (außer 0) abgelegt. In diesem Fall kann die Ein- oder Ausgabe von Daten im Zusammenhang mit der Diskettenstation oder dem Programmrecorder über den Lautsprecher des Fernsehgerätes akustisch mitver-

Tabelle G-2: Inhalt der Speicherzelle 53279 beim Druck einer oder mehrerer Funktionstasten

Inhalt der Speicherzelle 53279 (dezimal)	Gedrückte Funktionstaste(n)
0	OPTION, SELECT und START
1	OPTION und SELECT
2	OPTION und START
3	OPTION
4	SELECT und START
5	SELECT
6	START
7	Keine

folgt werden. Wenn hier jedoch eine 0 gespeichert ist, dann findet diese Tonerzeugung nicht statt.

Drucker

29 Zeiger des Druckerbuffers
(Printer Buffer Pointer = PBPNT)

Diese Speicherzelle enthält einen Zeiger zur Adressierung der Stelle des Druckerbuffers, auf die als nächstes zugegriffen wird. Der Inhalt dieser Speicherzelle kann zwischen 0 und dem Wert, der in PBUFSZ (Speicherzelle 30) abgelegt ist, liegen.

30 Größe des Druckerbuffers
(Printer Buffer Size = PBUFSZ)

Mit Hilfe dieser Speicherzelle wird die Größe des Druckerbuffers festgelegt. Hier ist eine 40 abgelegt, wenn normale Zeichen gedruckt werden sollen und eine 29, wenn liegende Zeichen gedruckt werden sollen.

960-999 Druckerbuffer
(Printer Buffer = PRNBUF)

An den Drucker ausgegebene Zeichen werden hier zwischengespeichert. Sobald ein EOL-Zeichen ausgegeben wird oder der Buffer voll ist, wird der gesamte Inhalt des Buffers an den Drucker übertragen.

Für den Anwender zur Verfügung stehende Speicherbereiche

1536-1663 Bedingt verfügbar
(Conditionally Available)

Normalerweise steht dieser Speicherbereich für Maschinenprogramme, Display Lists usw. zur Verfügung. Wenn jedoch bei der Ausführung eines INPUT-Befehls mehr als 128 Zeichen empfangen werden, dann werden diese Zeichen hier zwischengespeichert.

1664-1791 Immer verfügbar
(Unconditionally Available)

Dieser Speicherbereich steht immer für Maschinenprogramme, Display Lists usw. zur Verfügung.

186,187 Zeilennummer, in der ein Programm unterbrochen wurde (Stop Line Number = STOPLN)

Diese Speicherzellen enthalten die Zeilennummer eines BASIC-Programmes, in der das Programm durch Drücken der BREAK-Taste oder einen STOP-Befehl unterbrochen wurde oder in der ein Fehler auftrat.

195 Kodezahl des aufgetretenen Fehlers (Error Number = ERRSAV)

Wenn ein Fehler auftritt, wird die Kodezahl des Fehlers in dieser Speicherzelle abgelegt. In Teil A des Anhanges sind die Kodezahlen und die entsprechenden Fehlerursachen aufgeführt.

212,213 Funktionswert der USR-Funktion (USR Function Value = FR0)

In diesen Speicherzellen kann durch ein mit Hilfe der USR-Funktion aufgerufenes Maschinenprogramm ein numerischer Wert abgelegt werden, der beim Rücksprung in das BASIC-Programm als Funktionswert übergeben wird.

251 Grad/Bogenmaß-Umschaltung (Radians or Degrees = RADFLG oder DEGFLG)

Wenn hier eine 0 abgelegt ist, wird im Bogenmaß und wenn hier eine 6 abgelegt ist, in Grad gerechnet.

564,565 Position des Light Pen (Light Pen Position = LPENH und LPENV)

Der Inhalt der Speicherzelle 564 entspricht der horizontalen und der Inhalt der Speicherzelle 565 der vertikalen Position des Light Pen. Die Zeilen- und Spaltenaufteilung ist dabei allerdings nicht dieselbe wie die der gerade aktivierten Grafikbetriebsart. In horizontaler Richtung gibt es 228 mögliche Positionen des Light Pen. Die am weitesten links liegende Position trägt die Nummer 67. Jedesmal, wenn der Light Pen um eine Position nach rechts bewegt wird, wird der Inhalt der Speicherzelle 564 um 1 erhöht. Auf die Position 255 folgt die Position 0. Danach wird wieder schrittweise um 1 erhöht. Die am weitesten rechts liegende Position trägt die Nummer 7. In horizontaler Richtung gibt es 96 mögliche Positionen des Light Pen. Die oberste Position trägt die Nummer 16 und die unterste die Nummer 11.

Tabelle G-3: Zuordnung der einzelnen Bits der Speicherzelle 53744 zu den verschiedenen Interrupts

Bit	Interrupt
0	Timer 1
1	Timer 2
2	Timer 4
3	Serielle Datenübertragung (Byte-Ausgabe) beendet
4	Daten zur seriellen Ausgabe benötigt
5	Serielle Datenübertragung (Eingabe) beendet
6	Taste gedrückt (außer BREAK)
7	BREAK-Taste gedrückt

Interruptüberwachung

53744 Interruptstatus und -freigabe
(IRQ Interrupt Status/Enable = IRQST/IRQEN)

Mit Hilfe eines PEEK-Befehls kann der Interruptstatus festgestellt werden und mit Hilfe eines POKE-Befehls ein Interrupt freigegeben oder gesperrt werden. Jedes Bit ist für einen bestimmten Interrupt zuständig (siehe Tabelle G-3). Wenn der Inhalt dieser Speicherzelle gelesen wird und ein Bit 0 ist, dann ist der entsprechende Interrupt aktiv. Wenn hingegen ein Bit 1 ist, dann ist der entsprechende Interrupt nicht aktiv. Wenn ein Wert in dieser Speicherzelle abgelegt wird, dann bedeutet eine 0, daß der entsprechende Interrupt gesperrt und eine 1, daß der entsprechende Interrupt freigegeben ist.

Anhang H

TABELLEN ZUR ZAHLENUMWANDLUNG

Mit Hilfe der in diesem Abschnitt enthaltenen Tabellen können

- Hexadezimal – in Binärzahlen (und umgekehrt) und
- Hexadezimal – in Dezimalzahlen (und umgekehrt)

umgewandelt werden.
Mit Hilfe von Tabelle H-1 können Sie Hexadezimalzahlen von 00 bis 0F in Binärzahlen von 0000 bis 1111 umwandeln (und umgekehrt).
Wenn Sie größere Binärzahlen umwandeln wollen, nehmen Sie jeweils vier Stellen der Binärzahl und wandeln Sie in eine Hexadezimalzahl um. Dabei gehen Sie von rechts nach links vor. Wenn dabei am linken Ende der Zahl weniger als vier Stellen übrigbleiben, setzen Sie an den Anfang der Zahl einige zusätzliche Nullen. Das sieht dann zum Beispiel so aus:

$$100101_2 = \underbrace{\underbrace{00100101_2}_{2\quad 5}}_{25_{16}}$$

Wenn Sie eine Hexadezimalzahl, die größer als 0F ist, in eine Binärzahl umwandeln wollen, tun Sie das Stelle für Stelle, zum Beispiel so:

$$37_{16} \rightarrow 3 \quad 7 \rightarrow \underbrace{\underbrace{0110}_{3}\ \underbrace{0111}_{7}}_{01100111_2}$$

Tabelle H-1: Umwandlung von Hexadezimalzahlen in Binärzahlen und umgekehrt

Hexadezimal	Binär	Hexadezimal	Binär
00	0000	08	1000
01	0001	09	1001
02	0010	0A	1010
03	0011	0B	1011
04	0100	0C	1100
05	0101	0D	1101
06	0110	0E	1110
07	0111	0F	1111

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt

Mit Hilfe der untenstehenden Tabelle können ganzzahlige Hexadezimalzahlen von 0 bis FFF direkt in die entsprechende Dezimalzahl (von 0 bis 4095) umgewandelt werden (oder umgekehrt). Um größere Zahlen umzuwandeln, wird zusätzlich die folgende Tabelle benötigt:

Hexadecimal	Decimal	Hexadecimal	Decimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

Um einen Hexadezimalbruch in einen Dezimalbruch umzuwandeln, gehen Sie folgendermaßen vor:

1. Formen Sie den Hexadezimalbruch in eine ganze Zahl, multipliziert mit 16^{-n} um. Dabei entspricht n der Anzahl von Nachkommastellen des Hexadezimalbruchs.

$$0.\,CA9BF3_{16} = CA9\,BF3_{16} \times 16^{-6}$$

2. Wandeln Sie den ganzzahligen Teil der Hexadezimalzahl in eine Dezimalzahl um.

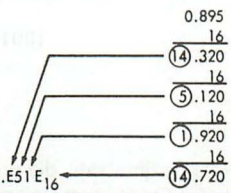
$$CA9\,BF3_{16} = 13\,278\,195_{10}$$

3. Multiplizieren Sie diese Dezimalzahl mit 16^{-n}

$$13\,278\,195 \times 596\,046\,448 \times 10^{-16} = 0.791\,442\,096_{10}$$

Dezimalbrüche können in Hexadezimalbrüche umgewandelt werden, indem der Dezimalbruch fortlaufend mit 16_{10} multipliziert wird. Nach jeder Multiplikation wird der ganzzahlige Teil des Ergebnisses abgetrennt, in eine Hexadezimalzahl umgewandelt (schließlich wurde bisher nur Dezimal gerechnet) und als Nachkommastelle einer Hexadezimalzahl eingesetzt.

Beispiel: Der Dezimalbruch $0{,}895_{10}$ soll in einen Hexadezimalbruch umgewandelt werden.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt (Fortsetzung)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt (Fortsetzung)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1562	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt (Fortsetzung)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt (Fortsetzung)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

Tabelle H-2: Umwandlung von Hexadezimalzahlen in Dezimalzahlen und umgekehrt (Fortsetzung)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	348º	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

Anhang I

LITERATURHINWEISE

BASIC

Albrecht, Finkel, and LeBaron. *What to Do After You Hit Return.* Rochelle Park, N.J.: Hayden Book Company.

Coan, James S. *Advanced BASIC.* Rochelle Park, N.J.: Hayden Book Company.

Coan, James S. *Basic BASIC.* Rochelle Park, N.J.: Hayden Book Company.

Dwyer, T., and Critchfield, Margot. *BASIC and the Personal Computer.* Reading, Mass.: Addison-Wesley, 1980.

Neirson, John M. *The Little Book of BASIC Style.* Reading, Mass.: Addison-Wesley, 1978.

Programmierung in Assemblersprache

DeJong, Marvin. *Programming and Interfacing the 6502, With Experiments.* Indianapolis: Howard W. Sams, 1980.

Foster, Caxton C. *Programming a Microcomputer: 6502.* Reading, Mass.: Addison-Wesley, 1978.

Leventhal, Lance A. *6502 Assembly Language Programming.* Berkeley: Osborne/McGraw-Hill, 1979.

Osborne, Adam. *An Introduction to Microcomputers: Volume 1 — Basic Concepts.* 2nd ed. Berkeley: Osborne/McGraw-Hill, 1980.

Scanlon, Leo J. *6502 Software Design.* Indianapolis: Howard W. Sams.

Zaks, Rodnay. *6502 Applications Book.* Berkeley: Sybex, 1979.

Zeitschriften

Atari Connection. 1265 Borregas Avenue, P.O. Box 427, Sunnyvale, California 94086.

BYTE. 70 Main Street, Peterborough, New Hampshire 03458.

Compute! P.O. Box 5406, Greensboro, North Carolina 27403.

Creative Computing. 39 East Hamover Avenue, Morris Plains, New Jersey 07950.

Desktop Computing. 80 Pine Street, Peterborough, New Hampshire 03458.

Micro. P.O. Box 6502, Chelmsford, Massachusetts 01824.

Microcomputing. 80 Pine Street, Peterborough, New Hampshire 03458.

Personal Computing. P.O. Box 13916, Philadelphia, Pennsylvania 19101.

Popular Computing. 70 Main Street, Peterborough, New Hampshire 03458.

Purser's Magazine. P.O. Box 466, El Dorado, California 95623.

Recreational Computing. 1263 El Camino Real, Menlo Park, California 94025.

ATARI-Veröffentlichungen

The following publications are available from Atari, Inc., 1265 Borregas Avenue, P.O. Box 427, Sunnyvale, California 94086.

Albrecht, Bob; Brown, Jerald R.; Finkel, LeRoy. *Atari BASIC.* New York, Chichester, Brisbane, Toronto: John Wiley & Sons, Inc., 1979.

ATARI 810 Disk Drive Operator's Manual.

ATARI 400/800 Disk Utility.

ATARI 400/800 Operating Systems.

ATARI 825 80-Column Printer Operator's Manual.

ATARI 850 Interface Module Operator's Manual.

ATARI Personal Computer System Operating Systems User's Manual and Hardware Manual.

ATARI 400/800 Basic Reference Manual.

ATARI 400/800 Disk Operating Systems Reference Manual.

Anhang J
NEUE ATARI-PRODUKTE

Sehr verehrter ATARI-Freund!

Zur internationalen Funkausstellung in Berlin im Herbst 1983 stellte ATARI eine völlig neue Produktserie vor. Erste Anzeichen dieser Veränderungen konnte man bereits auf der Hannover Messe im Frühjahr 1983 bemerken. Hier wurde nämlich der ATARI 1200 XL vorgestellt. Es handelte sich dabei um ein Gerät mit einem vollkommen neuen Design. Die Unterschiede zu den alten Geräten bestanden zum Beispiel in einer verbesserten Farbintensität, einem eingebauten Selbsttest des Computers, zusätzlichen Funktionstasten und einem integrierten internationalen Zeichensatz. Doch dieses Gerät ist nie auf den deutschen Markt gekommen.
Die neue Produktreihe von ATARI umfaßt insgesamt 3 neue Computer und 8 Peripherieeinheiten. Auf den nachfolgenden Seiten wollen wir Ihnen diese Geräte vorstellen und einige herausragende Besonderheiten aufzeigen.

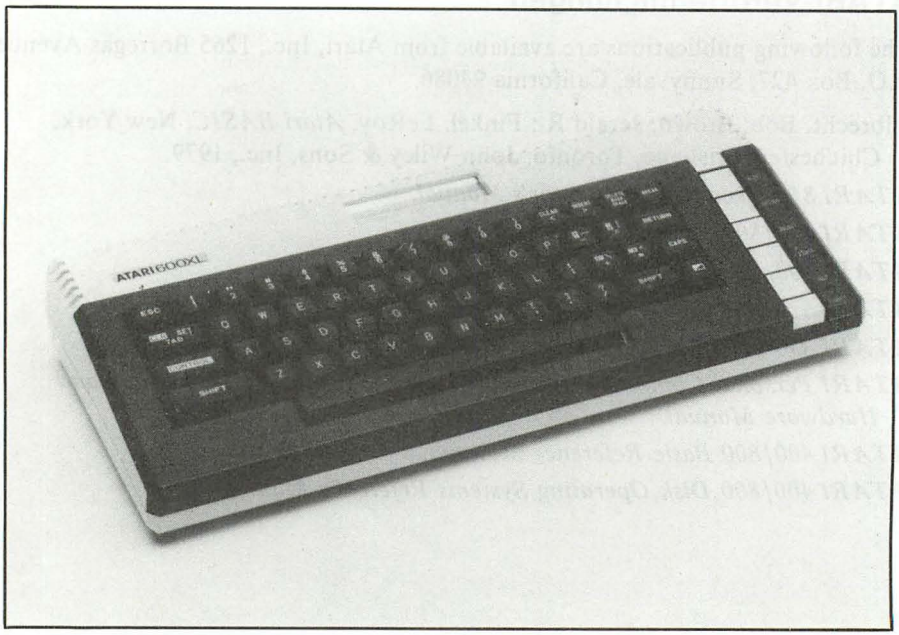


Bild J-1: ATARI 600 XL

Die neuen Computer

ATARI hat insgesamt 3 neue Computer auf den Markt gebracht. Es handelt sich dabei um die Modelle 600 XL, 800 XL und 1450 XLD. Gleichzeitig wurden die alten Modelle ATARI 400 und 800 aus der Produktreihe gestrichen. Der ATARI 600 XL kann als Ersatz für den ATARI 400 angesehen werden. Entsprechend ist der ATARI 800 XL dem alten Modell 800 zuzuordnen. Sehen wir uns nachfolgend die Geräte einmal an.

Der ATARI 600 XL

Der ATARI 600 XL ist mit einem 6502C Microprocessor ausgestattet. Die Programmiersprache BASIC ist fest im Betriebssystem integriert. Somit hat das Betriebssystem insgesamt 24K. Er ist in der Standardausführung mit 16K RAM (Random Access Memory = Arbeitsspeicher) ausgestattet. Mit einem Zusatzmodul kann er jedoch auch auf 64K RAM aufgerüstet werden. Er ist mit einer richtigen Schreibmaschinentastatur ausgestattet, im Gegensatz zur Folientastatur des alten Modelles ATARI 400. Mit der HELP-Taste stehen jetzt insgesamt 5 Funktionstasten zur Verfügung. Mit ihr sollen in bald zu erwartender Software Programminformationen aufgerufen werden können (bei den alten Geräten wird dies dann mit der ESC-Taste möglich sein). Ebenfalls integriert ist ein internationaler Zeichensatz. Er umfaßt sämtliche Sonderzeichen von verschiedenen Sprachen, wie zum Beispiel Französisch, Italienisch, Griechisch oder Schwedisch. Ganz selbstverständlich sind Ausstattungsdetails wie GTIA (Grafik-Display-Chip), Antic (Bildschirmkontroll- und I/O-Kontroll-Chip) oder Pokey (Tongenerator- und Steuerkontroll-Chip). Weiters sind nun 15 Grafik-Stufen direkt von Basic aus ansprechbar. Probleme, den ANTIC-Mode 7E anzusprechen, gehören der Vergangenheit an. Die Farbskala umfaßt 256 Farbabstufungen (davon können Sie 128 Abstufungen auf einmal auf dem Bildschirm erzeugen) in einer Auflösung von 320 x 192 Bildschirmpunkten. Die Farbintensität wurde drastisch erhöht – im Gegensatz zu den alten Modellen eine Steigerung von ca. 100 %. Weiterhin können Sie 4 völlig voneinander unabhängige Tongeneratoren mit einem Tonumfang über $3\frac{1}{2}$ Oktaven ansteuern. Für den Anschluß von Steuerknüppeln und ähnlichem stehen zwei Buchsen zur Verfügung (alte Software, die vier Buchsen ansprach, kann weiter verwendet werden, da intern umgeschaltet wird). Ein sehr wesentlicher Punkt ist die Möglichkeit, ein CPM-Erweiterungs-Modul anschließen zu können. Wollen Sie einmal wissen, ob Ihr Gerät noch in Ordnung ist, so können Sie einen Selbsttest aufrufen. Mit diesem Test können Sie den Speicher, die Tastatur und die Tongeneratoren selbst testen. Bei jedem Einschalten des Gerätes wird dieser Test automatisch ausgeführt. Sollte hierbei vom System eine Unregelmäßigkeit festgestellt werden, wird die Kontrolle nicht an BASIC weitergegeben, sondern automatisch der Selbsttest gestartet. Alles in allem handelt es sich beim neuen ATARI 600 XL um ein ideales Einsteigermodell.

Der ATARI 800 XL

Das Nachfolgermodell des ATARI 800 besitzt fest installierte 64K RAM. Auch er besitzt die zusätzliche HELP-Taste, Selbsttest und alle beim Modell 600 XL aufgeführten Details. Zusätzlich steht Ihnen noch ein Video/Audio-Direktanschluß zur Verfügung. Dieses Gerät bietet darüber hinaus die Möglichkeit, ein CPM-Erweiterungs-Modul anzuschließen. Somit kann erstmals CPM-Software auf einem ATARI-Computer verwendet werden.

Der ATARI 1450 XLD

Dieses Gerät ist im wahrsten Sinne des Wortes das Flagschiff der neuen ATARI-Produktreihe. Neben den eben beschriebenen Details der Geräte 600 XL und 800 XL bietet dieses Modell zwei wesentliche Erweiterungen. Der ATARI 1450 XLD besitzt eine integrierte Sprachsynthese. Sie können also Text nicht mehr nur auf dem Bildschirm oder auf einem Drucker ausgeben, sondern auch als gesprochene Stimme. Die Anschaffung eines separaten Sprachsyntesizers entfällt. Die zweite Erweiterung besteht aus einem eingebauten Double-Density-Doppellaufwerk. Dieses Laufwerk besitzt für bisherige ATARI-Verhältnisse sensationelle Speicherkapazität. Insgesamt können auf einer $5\frac{1}{4}$ Inch-Diskette nun 254 KByte gespeichert werden. Das Gehäuse über dem Laufwerk gestattet es, dort direkt einen Datenmonitor zu installieren. Dieses Gerät kann durchaus schon als Bürocomputer bezeichnet werden.

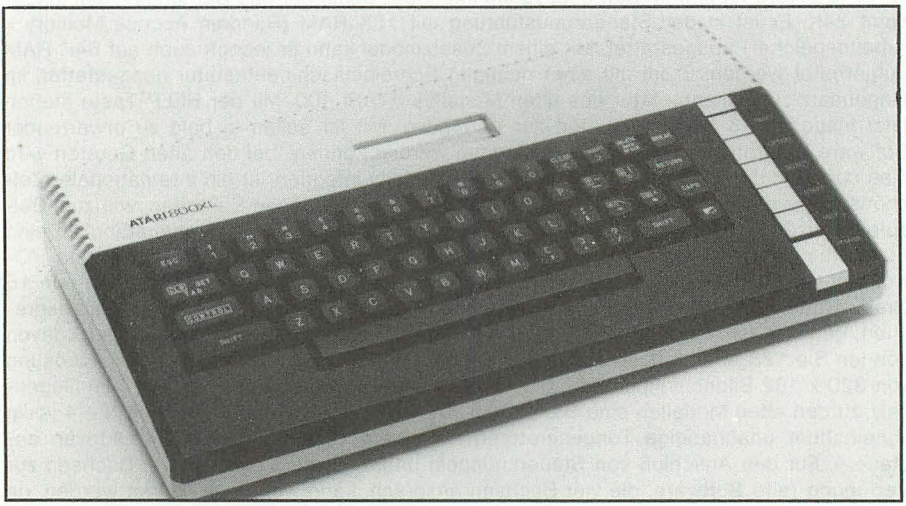


Bild J-2: ATARI 800 XL

Neue Peripherie

Mit den neuen Computern brachte ATARI auch 8 neue Peripherie-Einheiten auf den Markt. Es handelt sich dabei um drei Drucker, zwei Speichermedien und drei I/O-Kontrollgeräte. Nachfolgend wollen wir Ihnen diese Geräte vorstellen.

DRUCKER:

Der ATARI 1027-Typenraddrucker

Bei diesem Gerät handelt es sich um einen Typenraddrucker mit einer Druckgeschwindigkeit von 20 Zeichen je Sekunde. Sie können normales DIN A4-Papier verwenden. Die Qualität der Ausdrucke kann als korrespondenzfähig bezeichnet werden. Der Drucker kann direkt ohne zusätzliches Modem an die Computer angeschlossen werden. Insgesamt ist er der ideale Drucker für Geschäftsbriefe und andere Dokumente, die in optimaler Qualität gedruckt werden müssen.

Sie können auch Zeichen mit Unterlängen ausdrucken. Zum Einschalten dieses Modus drücken Sie zweimal die ESC-Taste. Drücken Sie dann CTRL und gleichzeitig die Taste mit dem Buchstaben Z. Der nachfolgende Text wird bis zum Ende der Zeile mit Unterlänge ausgedruckt. Mit einer erneuten Eingabe des eben beschriebenen Befehles können Sie den Ausdruck mit Unterlänge auch vor Zeilenende wieder abschalten.
Der Ausdruck mit Unterlänge kann auch über den Befehl (ESC)(ESC)(CTRL)(O) eingeschaltet und mit (ESC)(ESC)(CTRL)(N) wieder abgeschaltet werden.
Um mit dem Drucker auch Sonderzeichen drucken zu können, ist er mit einem internationalen Zeichensatz ausgestattet. Um diesen Zeichensatz zu aktivieren, geben Sie (ESC)(ESC)(CTRL)(W) ein. Jetzt können Sie alle Sonderzeichen ausdrucken lassen. Um diese Betriebsart wieder abzuschalten, geben Sie (ESC)(ESC)(CTRL)(X) ein.
Hier ein kleines Beispielprogramm zum Experimentieren:

```
10 OPEN #2,8,0,"P:"
20 PRINT #2;"(ESC)(ESC)(CTRL)(W)"
30 PRINT #2;"(beliebiger Text mit Sonderzeichen)"
40 PRINT #2;"(ESC)(ESC)(CTRL)(X)"
```

Drücken Sie bitte für die in Klammern dargestellten Werte die entsprechenden Tasten auf der Tastatur.

Befehl	Code	Hex-Code
CTRL ,	00	00
CTRL K	11	0B
CTRL Q	17	11
CTRL X	24	18
CTRL Y	25	19
CTRL Z	26	1A
CTRL ;	123	7B
CTRL D	04	04

Der ATARI 1025-Matrixdrucker

Bei diesem Druckermodell handelt es sich um einen Matrixdrucker mit einer Druckgeschwindigkeit von 40 Zeichen je Sekunde. Zum Drucken wird normales Endlospapier mit einer Breite von 12 Inch verwendet. Sie können innerhalb Ihrer Programme die Schriftarten des Druckers umstellen. Für einen schnellen Ausdruck eines Programmlistings oder auch eines Briefes ist dieser Drucker sehr gut zu verwenden. Als Farbband kann jedes Schreibmaschinen-Farbband genommen werden.
Mit verschiedenen Steuerzeichen können Sie den Seitenrand, die Schriftgröße und einen europäischen Zeichensatz anschalten. Die folgende Tabelle zeigt Ihnen einige der Ihnen zur Verfügung stehenden Funktionen und wie die Funktionen aufgerufen werden können.

Befehl	Kontroll-Kode (Hex)	Funktion
ESC, CTRL T	1B,14	Es werden 16,5 cpi gedruckt
ESC, CTRL O	1B,0F	Es werden 10 cpi gedruckt
ESC, CTRL N	1B,0E	Es werden 5 cpi gedruckt
ESC, L	1B,4C	Ergibt lange Zeile (80 Zeichen)
ESC, S	1B,53	Ergibt kurze Zeile (64 Zeichen)
ESC, 6	1B,36	Ergibt 6 LPI (^{1}/6" Leerstellen)
ESC, 8	1B,38	Ergibt 8 LPI (^{1}/8" Leerstellen)

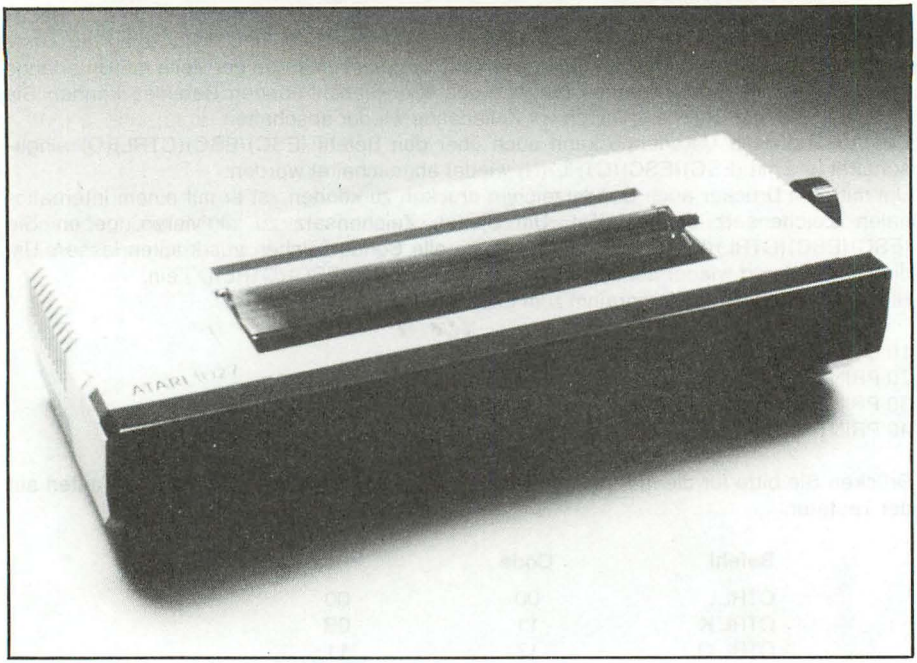


Bild J-3: Der ATARI 1027 Drucker

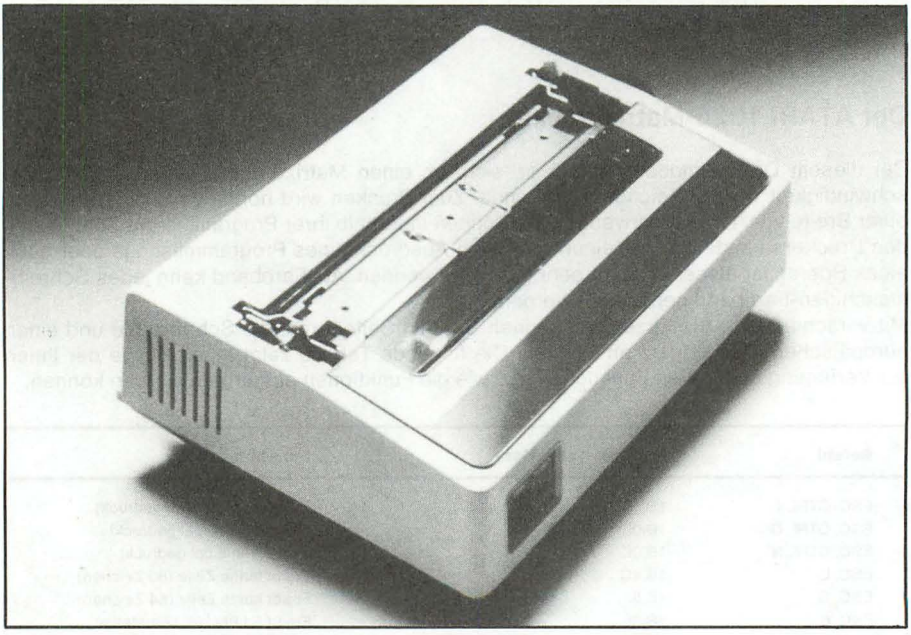


Bild J-4: Der ATARI 1025 Drucker

Der ATARI 1020-Farbdrucker

Dieser Drucker arbeitet mit einem 40-Zeichen Rollenpapier. Er kann vier Farben auf einmal drucken. Sie können also Farbgraphiken erstellen und ausdrucken lassen. Außerdem stehen Ihnen 64 verschiedene Schriftgrößen zur Verfügung. Die Ansteuerung dieses Drukkers ist problemlos innerhalb Ihrer Programme möglich. Dieser Drucker ist besonders für denjenigen zu empfehlen, der neben kurzen Ausdrucken auch Graphiken, zum Beispiel Funktionszeichnungen usw., anfertigen will.
Der Drucker arbeitet wahlweise entweder im Textmodus oder im Graphikmodus. Insgesamt stehen Ihnen 13 verschiedene Funktionen zur Verfügung. Um den Graphikmodus einzuschalten, müssen Sie folgende Befehle eingeben:

OPEN #(Wert zwischen 1 und 7),8,0,"P:"

Wir verwenden hier als Beispiel Datenkanal #2.

PRINT #2;"(ESC)(ESC)(CTRL)(A)"

Der Drucker befindet sich nun im Graphikmodus. Wenn Sie sich eine Befehlsliste ausdrucken lassen wollen, so geben Sie folgenden Befehl ein:

PRINT #;"(eckige Klammer zu)"

Wollen Sie wieder in den Textmodus zurückkehren, so geben Sie folgenden Befehl ein:

PRINT #2;"A"

Im Textmodus stehen Ihnen 3 verschiedene Zeichenbreiten zur Verfügung. Sie können entweder 20, 40 oder 80 Zeichen pro Zeile drucken. Nach dem Einschalten befindet sich das Gerät automatisch im Textmodus mit einer Zeichenbreite von 40 Zeichen pro Zeile. Um nur 20 Zeichen pro Zeile auszudrucken, geben Sie nachfolgenden Befehl ein:

LPRINT"(ESC)(ESC)(CTRL)(P)"

Wollen Sie jedoch 80 Zeichen je Zeile ausdrucken, so müssen Sie diesen Befehl verwenden:

LPRINT"(ESC)(ESC)(CTRL)(S)"

Um wieder in den normalen Modus von 40 Zeichen je Zeile zurückzukehren, geben Sie folgenden Befehl ein:

LPRINT"(ESC)(ESC)(CTRL)(N)"

Neben diesen Umschaltmöglichkeiten können Sie auch die Zeichengröße drastisch variieren. Insgesamt sind 64 verschiedene Größen möglich. Der Befehl nennt sich SET SCALE und wird folgendermaßen angesprochen:

PRINT#2;"S (und eine Zahl zwischen 0 und 63)"

Wenn Sie jetzt im Textmodus Text ausdrucken, so wird jedes Zeichen in der ausgewählten Größe dargestellt.

Auch der ATARI 1020 Farbdrucker ist mit einem internationen Zeichensatz ausgestattet. Somit können alle Sonderzeichen gedruckt werden.

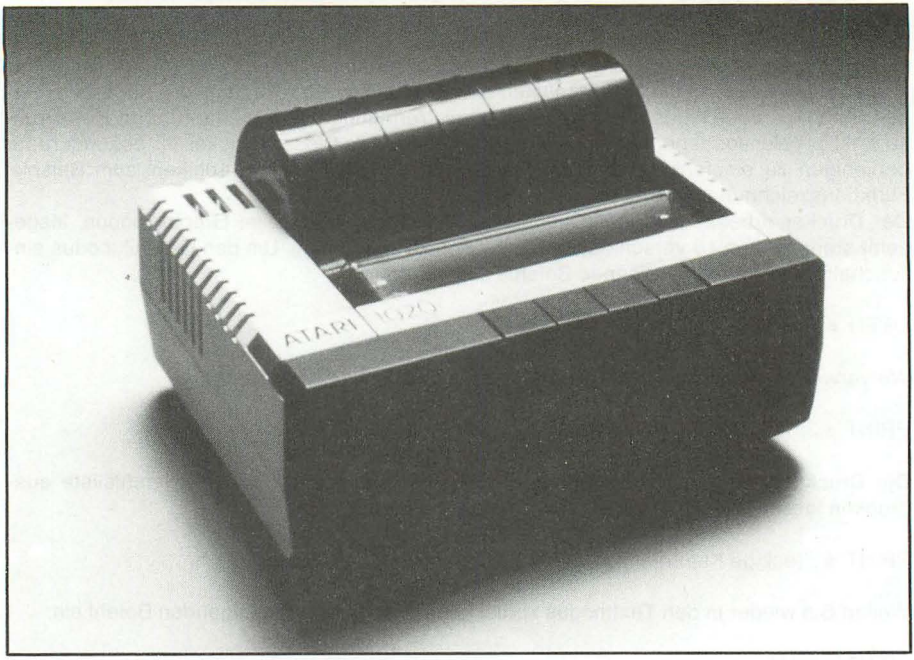


Bild J-5: ATARI 1020 Farbdrucker

Im Folgenden wollen wir Ihnen einen kurzen Überblick über den Graphikmodus und die Fähigkeiten des Druckers innerhalb dieser Funktion geben.
Insgesamt stehen Ihnen für Farbgraphiken 4 verschiedene Farben zur Verfügung. Um den Wagen mit dem Druckkopf in seine Ausgangsstellung zu bewegen, geben Sie folgenden Befehl ein:

```
PRINT#2;"H"
```

Zum Wechseln der einzelnen Farben geben Sie Folgendes ein:

```
PRINT #2;C (und eine Zahl zwischen 0 und 3)"
```

Sie können zwischen den Farben rot, schwarz, grün und blau auswählen. Die ist jedoch primär von den eingesetzten Farbstiften abhängig.

Nun zu den einzelnen Graphikbefehlen. Nach dem Einschalten des Druckers zeichnet dieser im Graphikmodus automatisch durchgehende Linien. Sie können aber Linien auch punktiert darstellen lassen. Dabei können Sie den Abstand zwischen den Punkten bestimmen. Geben Sie hierzu den folgenden Befehl ein:

```
PRINT #2;"L0"
```

Zum Zeichnen können Sie nun den DRAW-Befehl verwenden. Nach der Eingabe dieses Befehles zeichnet der Drucker eine Linie von der Koordinate 0,0 zur Koordinate X,Y. Die Werte für X und Y müssen von Ihnen bestimmt werden. Der Wert von X muß im Bereich von

0 und 480, der Wert für Y zwischen –999 und 999 liegen. Zur Eingabe dieses Befehles verfahren Sie wie folgt:

PRINT #2;"DX,Y"

Sie können auch mehrere Befehle eingeben:

PRINT #2;"DX,Y;X,Y;X,Y;X,Y;"

Die Koordinate 0,0 wird durch den folgenden Befehl initialisiert:

PRINT #2;"I"

Weiterhin stehen Ihnen ein RELATIVE DRAW-, MOVE- und RELATIVE MOVE-Befehl zur Verfügung. Sie dienen sämtlich der Steuerung des Druckkopfes zu den einzelnen zu zeichnenden Punkten und Linien.
Außerdem können Sie mit dem Befehl AXIS DRAWING (Achsen Zeichnen) X- und Y-Achsen zeichnen lassen. Der Befehl lautet wie folgt:

PRINT #2;"X1,10,50"

Mit X1 bestimmen Sie, daß es sich um eine X-Achse handelt. Mit der 10 legen Sie den Abstand der Einzelmarkierungen auf der Achse fest. Mit 50 bestimmen Sie die Anzahl der Markierungen auf der Achse. Um eine Y-Achse zu zeichnen, müssen Sie X1 gegen X0 ersetzen.
Wollen Sie nun die Achsen beschriften, so können Sie hierzu sogar die Schrift in der entsprechenden Lage drucken. Diese Funktion wird als ALPHA ROTATE-Modus bezeichnet. Der Befehl lautet:

PRINT #2;"Q (und eine Zahl zwischen 0 und 3)"

Für einen normalen Textausdruck innerhalb des Graphikmodus steht Ihnen der folgende Befehl zur Verfügung:

PRINT #2;"P (und der entsprechende Text)"

Mit dem P wird dem Drucker mitgeteilt, die nun folgenden alphanumerischen Zeichen auszudrucken.

Zusammenfassend handelt es sich beim ATARI 1020 Farbdrucker um ein Gerät, mit dem Ihren künstlerischen Fähigkeiten fast keine Grenzen gesetzt sind. Die Druckgeschwindigkeit ist mit 10 Zeichen pro Sekunde nicht sehr schnell, aber die vielen Optionen dürften diesen "Nachteil" aufwiegen.
Hier haben wir Ihnen nun noch ein kleines Testprogramm vorbereitet, das einen netten Graphikausdruck ergibt.

```
10 DIM D(11,2)
20 OPEN #2,8,0,"P:"
30 PRINT #2;"ö";PRINT #2;"C0"
40 PRINT #2;"P  *****PLOTTFÄHIGKEITEN*****"
50 PRINT #2;"H";PRINT #2;"M240,-200"
60 PRINT #2;"I";PRINT#2;"C0"
80 D(I,0)=SIN(S)*110
90 D(I,I)=COS(S)*110
100 NEXT I
```

```
110 C=2
120 PRINT #2;"M";D(0,0);",";D(0,1)
130 FOR I=1 TO 5
140 C=C+I:IF C>3 THEN C=0
150 PRINT #2;"C";C:K=0
160 FOR J=1 TO 11
170 K=K+I:IF K>10 THEN K=K-11
180 PRINT #2;"D";D(K,0);",";D(K,I)
190 NEXT J
200 NEXT I
210 PRINT #2;"H"
220 PRINT #2;"M0,-240"
230 PRINT #2;"A"
240 CLOSE #2
250 END
```

SPEICHERMEDIEN:

Der ATARI 1010 Programm-Recorder

Bei diesem Gerät handelt es sich um das kostengünstigste Massenspeichermedium. Sie können jede beliebige im Handel erhältliche Leerkassette verwenden. Auf einer normalen C60 Leerkassette können Sie somit insgesamt ca. 100 KByte abspeichern. Außerdem besteht die Möglichkeit, auf der zweiten Tonspur des Bandmaterials ein Programm mit gesprochenen Texten oder mit Musik zu unterlegen.

Der ATARI 1050 Dual-Density Disk Drive

Mit diesem Gerät haben Sie ein Speichermedium, das schnellen Zugriff zu Daten und Programmen ermöglicht. Das Laufwerk arbeitet mit normalen $5\frac{1}{4}$ Inch-Disketten. Mit DOS 2 können Sie auf einer Diskette 88 KByte abspeichern. Dies entspricht genau der Speicherkapazität der alten Diskettenstationen. Sie können bis zu vier dieser Laufwerke an einen ATARI-Computer anschließen. Das Design des Gerätes ist, wie die gesamte neue Gerätegeneration von ATARI, im Stil der SLIM LINE Linie. In Kürze wird auch ein neues Disketten-Betriebssystem erhältlich sein (DOS III). Mit diesem DOS werden dann 127 KByte auf einer Diskette zu speichern sein. Es stehen Ihnen dann 1024 Sektoren (gegenüber 707 bei den alten Modellen) mit jeweils 128 BYTES zur Verfügung.

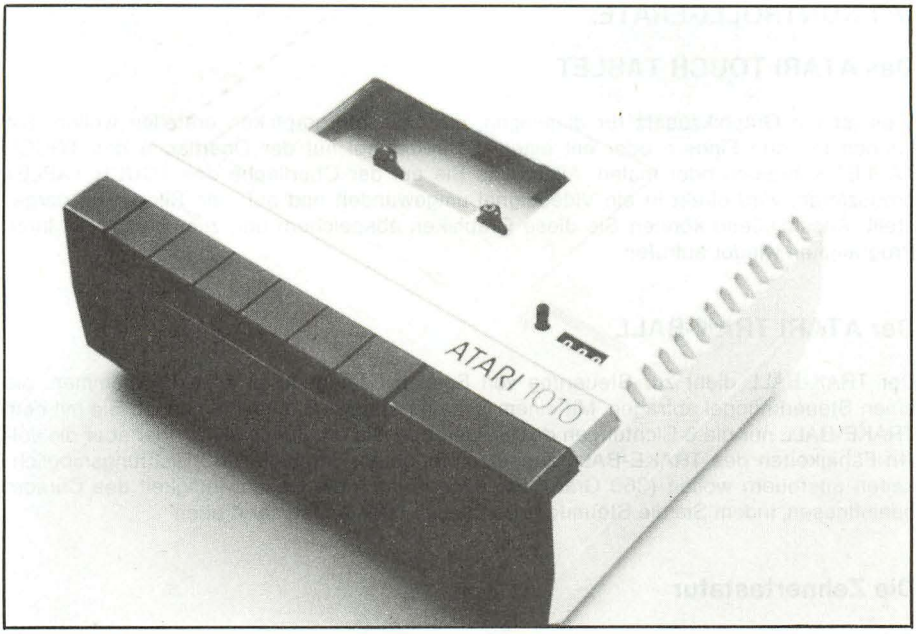


Bild 6: ATARI 1010 Programmrecorder

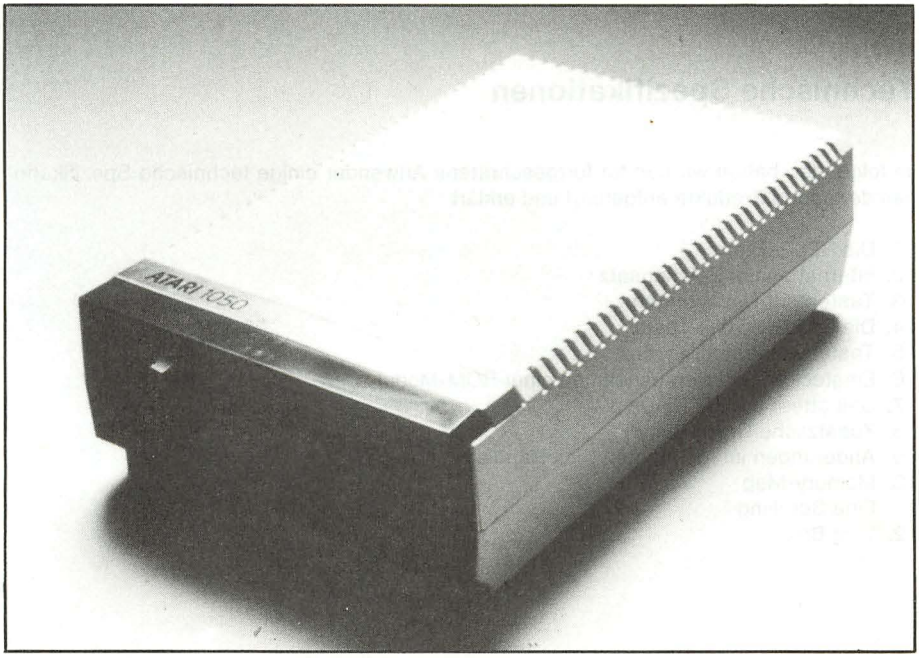


Bild 7: ATARI 1050 Disk Drive

I/O-KONTROLLGERÄTE:

Das ATARI TOUCH TABLET

Dies ist ein Graphikzusatz für diejenigen, die Computergraphiken erstellen wollen. Sie können mit den Fingern oder mit einem Schreibgriffel auf der Oberfläche des TOUCH TABLET schreiben oder malen. Alles, was Sie auf der Oberfläche des TOUCH TABLET produzieren, wird direkt in ein Videosignal umgewandelt und auf dem Bildschirm dargestellt. Anschließend können Sie diese Graphiken abspeichern und zum Beispiel in Ihren Programmen wieder aufrufen.

Der ATARI TRAK-BALL

Der TRAK-BALL dient zur Steuerung von Spielen oder generell allen Programmen, die einen Steuerknüppel abfragen. Mit einem Schalter können Sie bestimmen, ob Sie mit dem TRAKE-BALL nur die 8 Richtungen des Steuerknüppels erhalten wollen, oder aber die vollen Fähigkeiten des TRAKE-BALL ausnutzen möchten, das heißt 360 Richtungsmöglichkeiten ansteuern wollen (360 Grad). Sie können auch die Geschwindigkeit des Cursors beeinflussen, indem Sie die Steuerkugel schneller oder langsamer drehen.

Die Zehnertastatur

Bei dieser Zahlentastatur handelt es sich um ein sehr nützliches Gerät, wenn man mit großen Zahlenmengen arbeiten muß. Zum Beispiel kann sie bei der Arbeit mit Visicalc sehr nützlich sein. Außerdem beinhaltet diese Tastatur weitere Funktionstasten zur Programmsteuerung.

Technische Spezifikationen

Im folgenden haben wir nun für fortgeschrittene Anwender einige technische Spezifikationen der neuen Produkte aufgeführt und erklärt.

1. Die HELP-Taste
2. Internationaler Zeichensatz
3. Tastenfeld-Umlegungen
4. Die CAPS-LOWR-Taste
5. Tastenwiederholfrequenz
6. Einstecken und Herausnehmen einer ROM-Modul
7. Selbsttest und BYTE-Display
8. Zusätzliche Graphikmodi
9. Änderungen im resistenten Disk Handler
10. Memory-Map
11. Fine Scrolling
12. Vioc Box

1. Die HELP-Taste

Während das Betriebssystem die Tastatur abfragt, stellt es auch fest, ob die Taste HELP gedrückt wurde. Die HELP-Taste besitzt im Betriebssystem als Flagge die Adresse $02DC bzw. 732.
Soll die HELP-Taste in einem Programm abgefragt werden, so genügt es nicht nur die Tastatur abzufragen. Man muß auch die Adresse $02DC abfragen. Dabei enthält diese Adresse folgende Werte:

HEX DEC Wert	Voraussetzungen
00 000	Die Adresse ist gelöscht. D. h. die HELP-Taste ist nicht gedrückt. Beim Einschalten des Computers wird diese Adresse gelöscht. Bevor man diese Adresse abfragt, sollte man sie löschen.
11 017	Es wurde nur die HELP-Taste gedrückt.
51 081	SHIFT und HELP wurden gedrückt.
91 145	CONTROL und HELP wurden gedrückt.

Wichtig: Soll die HELP-Taste mehrere Male in einem Programm abgefragt werden, so muß nach der Abfrage dieser Adresse in diese Adresse eine 0 gelegt werden.
Die HELP-Taste wird auch zum Einschalten des Selbst-Testes benötigt. Nähere Informationen dazu unter Selbst-Test.

2. Internationaler Zeichensatz

Wie bei den alten Geräten, so ist es auch bei den neuen Geräten möglich, einen Zeichensatz neu zu definieren. Die Basis des Zeichensatzes wird in der Adresse $02F4 bzw. 756 abgelegt. In dieser Adresse steht normalerweise $E0 bzw. 224. Die Angabe erfolgt in Seiten, d. h. die Zeichenbasis errechnet sich wie folgt:

Zeichenbasis = PEEK(756) * 256

Der normale Zeichensatz befindet sich demnach im ROM-Bereich von $E000 – $E3FF, da ja ein kompletter Zeichensatz aus 1024 Byte besteht. Der eingebaute internationale Zeichensatz befindet sich im ROM-Bereich von $CC00 – $CFFF. Um diesen anzusprechen, muß man den Inhalt der Adresse $02F4 auf $CC bzw. 756 auf 204 setzen. Die internationalen Zeichen sind über die Control zu erreichen.
Und nun noch die einzelnen ATASCII-Zeichen und ihre hexadezimalen und dezimalen ATASCII-Werte

á	00	0	♥	00	0	F	46	70	F	46	70						
ù	01	1	├	01	1	G	47	71	G	47	71						
ñ	02	2	│	02	2	H	48	72	H	48	72						
é	03	3	┘	03	3	I	49	73	I	49	73						
ç	04	4	┤	04	4	J	4A	74	J	4A	74						
ô	05	5	┐	05	5	K	4B	75	K	4B	75						
ò	06	6	/	06	6	L	4C	76	L	4C	76						
ì	07	7	\	07	7	M	4D	77	M	4D	77						
£	08	8	◢	08	8	N	4E	78	N	4E	78						
ï	09	9	▪	09	9	O	4F	79	O	4F	79						
ü	0A	10	◣	0A	10	P	50	80	P	50	80						
ä	0B	11	▫	0B	11	Q	51	81	Q	51	81						
ö	0C	12	▪	0C	12	R	52	82	R	52	82						
ú	0D	13	¯	0D	13	S	53	83	S	53	83						
ó	0E	14	_	0E	14	T	54	84	T	54	84						
ö	0F	15	▫	0F	15	U	55	85	U	55	85						
ü	10	16	♣	10	16	V	56	86	V	56	86						
ß	11	17	┌	11	17	W	57	87	W	57	87						
û	12	18	─	12	18	X	58	88	X	58	88						
î	13	19	┼	13	19	Y	59	89	Y	59	89						
é	14	20	●	14	20	Z	5A	90	Z	5A	90						
è	15	21	▬	15	21	[	5B	91	[	5B	91						
ñ	16	22	│	16	22	\	5C	92	\	5C	92						
ë	17	23	┬	17	23	]	5D	93	]	5D	93						
à	18	24	┴	18	24	^	5E	94	^	5E	94						
à	19	25	▌	19	25	_	5F	95	_	5F	95						
Á	1A	26	└	1A	26	`	60	96	♦	60	96						
Ę	1B	27	Ę	1B	27	a	61	97	a	61	97						
↑	1C	28	↑	1C	28	b	62	98	b	62	98						
↓	1D	29	↓	1D	29	c	63	99	c	63	99						
←	1E	30	←	1E	30	d	64	100	d	64	100						
→	1F	31	→	1F	31	e	65	101	e	65	101						
	20	32		20	32	f	66	102	f	66	102						
!	21	33	!	21	33	g	67	103	g	67	103						
"	22	34	"	22	34	h	68	104	h	68	104						
#	23	35	#	23	35	i	69	105	i	69	105						
$	24	36	$	24	36	j	6A	106	j	6A	106						
%	25	37	%	25	37	k	6B	107	k	6B	107						
&	26	38	&	26	38	l	6C	108	l	6C	108						
'	27	39	'	27	39	m	6D	109	m	6D	109						
(	28	40	(	28	40	n	6E	110	n	6E	110						
)	29	41	)	29	41	o	6F	111	o	6F	111						
*	2A	42	*	2A	42	p	70	112	p	70	112						
+	2B	43	+	2B	43	q	71	113	q	71	113						
,	2C	44	,	2C	44	r	72	114	r	72	114						
-	2D	45	-	2D	45	s	73	115	s	73	115						
.	2E	46	.	2E	46	t	74	116	t	74	116						
/	2F	47	/	2F	47	u	75	117	u	75	117						
0	30	48	0	30	48	v	76	118	v	76	118						
1	31	49	1	31	49	w	77	119	w	77	119						
2	32	50	2	32	50	x	78	120	x	78	120						
3	33	51	3	33	51	y	79	121	y	79	121						
4	34	52	4	34	52	z	7A	122	z	7A	122						
5	35	53	5	35	53	Ä	7B	123	♠	7B	123						
6	36	54	6	36	54	|	7C	124	|	7C	124						
7	37	55	7	37	55	Ñ	7D	125	Ñ	7D	125						
8	38	56	8	38	56	◀	7E	126	◀	7E	126						
9	39	57	9	39	57	▶	7F	127	▶	7F	127						
:	3A	58	:	3A	58	Ä	80	128	Ç	80	128						
;	3B	59	;	3B	59	Ö	81	129	▚	81	129						
<	3C	60	<	3C	60	Ü	82	130	▌	82	130						
=	3D	61	=	3D	61	É	83	131	▟	83	131						
>	3E	62	>	3E	62	ä	84	132	▐	84	132						
?	3F	63	?	3F	63	à	85	133	▜	85	133						
@	40	64	@	40	64	å	86	134	▞	86	134						
A	41	65	A	41	65	ç	87	135	▚	87	135						
B	42	66	B	42	66	ê	88	136	◤	88	136						
C	43	67	C	43	67	ë	89	137	▛	89	137						
D	44	68	D	44	68	è	8A	138	◥	8A	138						
E	45	69	E	45	69	ï	8B	139	▙	8B	139						
						ô	8C	140	▟	8C	140						

Ì	8D	141
Ä	8E	142
Å	8F	143
É	90	144
æ	91	145
Æ	92	146
ô	93	147
ö	94	148
ò	95	149
û	96	150
ù	97	151
ÿ	98	152
Ö	99	153
Ü	9A	154
¢	9B	155
£	9C	156
¥	9D	157
₧	9E	158
ƒ	9F	159
á	A0	160
í	A1	161
ó	A2	162
ú	A3	163
ñ	A4	164
Ñ	A5	165
ª	A6	166
º	A7	167
¿	A8	168
⌐	A9	169
¬	AA	170
½	AB	171
¼	AC	172
¡	AD	173
«	AE	174
»	AF	175
░	B0	176
▒	B1	177
▓	B2	178
│	B3	179
┤	B4	180
╡	B5	181
╢	B6	182
╖	B7	183
╕	B8	184
╣	B9	185
║	BA	186
╗	BB	187
╝	BC	188
╜	BD	189
╛	BE	190
┐	BF	191
└	C0	192
┴	C1	193
┬	C2	194
├	C3	195
─	C4	196
┼	C5	197
╞	C6	198
╟	C7	199
╚	C8	200
╔	C9	201
╩	CA	202
╦	CB	203
╠	CC	204
═	CD	205
╬	CE	206
╧	CF	207
╨	D0	208
╤	D1	209
╥	D2	210

Ì	8D	141
Ä	8E	142
Å	8F	143
É	90	144
æ	91	145
Æ	92	146
ô	93	147
ö	94	148
ò	95	149
û	96	150
ù	97	151
ÿ	98	152
Ö	99	153
Ü	9A	154
¢	9B	155
£	9C	156
¥	9D	157
₧	9E	158
ƒ	9F	159
á	A0	160
í	A1	161
ó	A2	162
ú	A3	163
ñ	A4	164
Ñ	A5	165
ª	A6	166
º	A7	167
¿	A8	168
⌐	A9	169
¬	AA	170
½	AB	171
¼	AC	172
¡	AD	173
«	AE	174
»	AF	175
░	B0	176
▒	B1	177
▓	B2	178
│	B3	179
┤	B4	180
╡	B5	181
╢	B6	182
╖	B7	183
╕	B8	184
╣	B9	185
║	BA	186
╗	BB	187
╝	BC	188
╜	BD	189
╛	BE	190
┐	BF	191
└	C0	192
┴	C1	193
┬	C2	194
├	C3	195
─	C4	196
┼	C5	197
╞	C6	198
╟	C7	199
╚	C8	200
╔	C9	201
╩	CA	202
╦	CB	203
╠	CC	204
═	CD	205
╬	CE	206
╧	CF	207
╨	D0	208
╤	D1	209
╥	D2	210

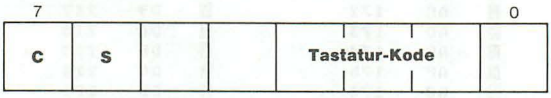
╙	D3	211
╘	D4	212
╒	D5	213
╓	D6	214
╫	D7	215
╪	D8	216
┘	D9	217
┌	DA	218
█	DB	219
▄	DC	220
▌	DD	221
▐	DE	222
▀	DF	223
α	E0	224
ß	E1	225
Γ	E2	226
π	E3	227
Σ	E4	228
σ	E5	229
µ	E6	230
τ	E7	231
Φ	E8	232
Θ	E9	233
Ω	EA	234
δ	EB	235
∞	EC	236
φ	ED	237
ε	EE	238
∩	EF	239
≡	F0	240
±	F1	241
≥	F2	242
≤	F3	243
⌠	F4	244
⌡	F5	245
÷	F6	246
≈	F7	247
°	F8	248
∙	F9	249
·	FA	250
√	FB	251
ⁿ	FC	252
²	FD	253
■	FE	254
⌂	FF	255

╙	D3	211
╘	D4	212
╒	D5	213
╓	D6	214
╫	D7	215
╪	D8	216
┘	D9	217
┌	DA	218
█	DB	219
▄	DC	220
▌	DD	221
▐	DE	222
▀	DF	223
α	E0	224
ß	E1	225
Γ	E2	226
π	E3	227
Σ	E4	228
σ	E5	229
µ	E6	230
τ	E7	231
Φ	E8	232
Θ	E9	233
Ω	EA	234
δ	EB	235
∞	EC	236
φ	ED	237
ε	EE	238
∩	EF	239
≡	F0	240
±	F1	241
≥	F2	242
≤	F3	243
⌠	F4	244
⌡	F5	245
÷	F6	246
≈	F7	247
°	F8	248
∙	F9	249
·	FA	250
√	FB	251
ⁿ	FC	252
²	FD	253
■	FE	254
⌂	FF	255

3. Tastenfeld-Umbelegungen

Mit den neuen Geräten ist es möglich, sich sein eigenes Tastenfeld zu erstellen. Es gibt nur wenige Tasten, die sich nicht umlegen lassen. Wollen wir uns zuerst mit dem Tabellenformat befassen:
Die Tastatur-Definitions-Tabelle besteht aus 192 Bytes und teilt sich in drei Blöcke auf.

64 Byte können mit 6 Bit-Stellen dargestellt werden. Es bleiben nun zu einer 8 Bit-Zahl noch 2 Bit übrig. Diese beiden Bits dienen zur Unterscheidung, ob CONTROL, SHIFT, CONTROL & SHIFT oder keine dieser Tasten gedrückt wurde. Diesen Wert, den man dann erhält, kann man in der Adresse $02FC bzw. 764 ablesen.
Dabei setzt sich der Inhalt dieser Adresse folgendermaßen zusammen.



C = CONTROL wurde gedrückt
S = SHIFT wurde gedrückt

Man darf nicht vergessen, daß der ATASCII-Wert erst noch gefunden werden muß. Hier ist nur von dem Tastatur-Kode die Rede. Über ihn wird der eigentliche ATASCII-Wert erst ermittelt.
Um sich ein eigenes Tastenfeld zu erstellen, muß man diesen internen Tasten-Kode kennen. Hier die Tabelle, in der die Tasten-Kodes mit den dazugehörigen ATASCII-Werten aufgelistet sind. In dieser Reihenfolge sind sie auch in den entsprechenden Blöcken im Betriebssystem abgelegt.

Tastatur Kode	Taste	Capslowr	Shift	Control
00 (00)	L	6C (108)	4C (076)	0C (012)
01 (01)	J	6A (106)	4A (074)	0A (010)
02 (02)	;	3B (059)	3A (058)	7B (123)
03 (03)	----	--- (---)	--- (---)	--- (---)
04 (04)	----	--- (---)	--- (---)	--- (---)
05 (05)	K	6B (107)	4B (075)	0B (011)
06 (06)	+	2B (043)	5C (092)	1E (030)
07 (07)	*	2A (042)	5E (094)	1F (031)
08 (08)	O	6F (111)	4F (079)	0F (015)
09 (09)	----	--- (---)	--- (---)	--- (---)

0A (10)	P	70 (112)	50 (080)	10 (016)
0B (11)	U	75 (117)	55 (085)	15 (021)
0C (12)	RET	9B (155)	9B (155)	9B (155)
0D (13)	I	69 (105)	49 (073)	09 (009)
0E (14)	-	2D (045)	5F (095)	1C (028)
0F (15)	=	3D (061)	7C (124)	1D (029)
10 (16)	V	76 (118)	56 (086)	16 (022)
11 (17)	HELP	-- (---)	-- (---)	-- (---)
12 (18)	C	63 (099)	43 (067)	03 (003)
13 (19)	---	-- (---)	-- (---)	-- (---)
14 (20)	---	-- (---)	-- (---)	-- (---)
15 (21)	B	62 (098)	42 (066)	02 (002)
16 (22)	X	78 (120)	58 (088)	18 (024)
17 (23)	Z	7A (122)	5A (090)	1A (026)
18 (24)	4	34 (052)	24 (036)	-- (---)
19 (25)	---	-- (---)	-- (---)	-- (---)
1A (26)	3	33 (051)	23 (035)	9B* (155)
1B (27)	6	36 (054)	26 (038)	-- (---)
1C (28)	ESC	1B (027)	1B (027)	1B (027)
1D (29)	5	35 (053)	25 (037)	-- (---)
1E (30)	2	32 (050)	22 (034)	FD (253)
1F (31)	1	31 (049)	21 (033)	-- (---)
20 (32)	,	2C (044)	5B (091)	00 (000)
21 (33)	SPACE	20 (032)	20 (032)	20 (032)
22 (34)	.	2E (046)	5D (093)	60 (096)
23 (35)	N	6E (110)	4E (078)	0E (014)
24 (36)	---	-- (---)	-- (---)	-- (---)
25 (37)	M	6D (109)	4D (077)	0D (013)
26 (38)	/	2F (047)	3F (063)	-- (---)
27 (39)	ATARI	-- (---)	-- (---)	-- (---)
28 (40)	R	72 (114)	52 (082)	12 (018)
29 (41)	---	-- (---)	-- (---)	-- (---)
2A (42)	E	65 (101)	45 (069)	05 (005)
2B (43)	Y	79 (121)	59 (089)	19 (025)
2C (44)	TAB	7F (127)	9F (159)	9E (158)
2D (45)	T	74 (116)	54 (084)	14 (020)
2E (46)	W	77 (119)	57 (087)	17 (023)
2F (47)	Q	71 (113)	51 (081)	11 (017)
30 (48)	9	39 (057)	28 (040)	-- (---)
31 (49)	---	-- (---)	-- (---)	-- (---)
32 (50)	0	30 (048)	29 (041)	-- (---)
33 (51)	7	37 (055)	27 (039)	-- (---)
34 (52)	BACKS	7E (126)	9C (156)	FE (254)
35 (53)	8	38 (056)	40 (064)	-- (---)
36 (54)	<	3C (060)	7D (125)	7D (125)
37 (55)	>	3E (062)	9D (157)	FF (255)
38 (56)	F	66 (102)	46 (070)	06 (006)
39 (57)	H	68 (104)	48 (072)	08 (008)
3A (58)	D	64 (100)	44 (068)	04 (004)
3B (59)	---	-- (---)	-- (---)	-- (---)
3C (60)	CAPS	-- (---)	-- (---)	-- (---)
3D (61)	G	67 (103)	47 (071)	07 (007)
3E (62)	S	73 (115)	53 (083)	13 (019)
3F (63)	A	61 (097)	41 (065)	01 (001)

* (CONTROL) 3 ist nicht mit **RETURN** gleichzusetzen. Es bewirkt ein **END OF FILE** kurz **EOF.**

Die inversen Zeichen werden nicht durch diesen Tastaturkode festgelegt. Soll ein Zeichen invers dargestellt werden, so wird über das entsprechende Zeichen eine Maske gelegt, die dann erst die Invertierung bewirkt.
Will man sein eigenes Tastenfeld zusammenstellen, so sollte man in dieser oben angeführten Tabelle gezielt Werte austauschen (bitte nicht die SHIFT- und CONTROL-Blöcke verwenden). Hat man sich sein eigenes Tastenfeld zusammengestellt, so legt man es im Speicher in folgender Reihenfolge ab. Zuerst die 64 Bytes Kleinbuchstaben, dann die 64 Bytes Großbuchstaben und dann die 64 Bytes Kontrollbuchstaben. Um dem Betriebssystem zu sagen, wo die neue Tabelle liegt, poket man in die Adresse $79,$7A bzw. 121,122 die Startadresse der Tabelle. Dabei geht man wie folgt vor:

```
HIGH=INT (STARTADRESSE)-256  (HIGH BYTE)
LOW=STARTADRESSE-A*256  (LOW BYTE)
POKE 121,LOW:POKE 122, HIGH
```

Wie bereits erwähnt gibt es nur wenige Tasten, die sich nicht umlegen lassen.

BREAK	Diese Taste hat eine bestimmte Adresse im OS
SHIFT	Dies ist eine Teilfunktion zur Bestimmung des Tasten-Kodes
CONTROL	Dies ist eine Teilfunktion zur Bestimmung des Tasten-Kodes
OPTION, SELECT START	Diese Tasten zählen nicht zur Tastatur
RESET	Diese Taste zählt nicht zur Tastatur
HELP	Hat eine eigene Adresse im OS
CTRL-1	Hat eine eigene Adresse im OS

4. Die CAPS-LOWR

Die CAPS-LOWR-Taste erfüllt bei den neuen Geräten eine etwas andere Funktion. Die Funktionsweise läßt sich am besten in einer Tabelle darstellen:

Tastenkombination	Vorher	Nachher
CAPS	Kontrollbuchstaben	Kleinbuchstaben
CAPS	Großbuchstaben	Kleinbuchstaben
CAPS	Kleinbuchstaben	Großbuchstaben
SHIFT-CAPS	—	Großbuchstaben
CTRL-CAPS	—	Kontrollbuchstaben
CTRL-SHIFT-CAPS	—	Keine Änderung

5. Tastenwiederholfrequenz

Das neue Betriebssystem bietet die Möglichkeit, die Wiederholfrequenz und die Zeitspanne, die es dauert bis die Wiederholfrequenz einsetzt, zu bestimmen. Die Wiederholfrequenz wird in der Adresse $02DA bzw. 730 festgelegt. Im Vertical-Blank wird diese Adresse heruntergezählt. Immer wenn diese Adresse eine 0 enthält, wird die Taste, die gerade gedrückt ist, noch einmal auf dem Bildschirm ausgegeben. Für NTSC (60 Hz) ist der Inhalt dieser Adresse 6, für PAL (50 Hz) 5. Dies bedeutet, daß für beide Fernsehnormen 10 Zeichen durch die Tastenwiederholfrequenz auf dem Bildschirm ausgegeben werden. Weiterhin bedeutet es, daß für NTSC das Maximum an ausgegebenen Zeichen bei 60–, bei PAL bei 50 Zeichen/Sekunde. Die Wiederholfrequenz errechnet sich wie folgt:

```
Wiederholfrequenz=(PAL=50/NTSC=60)PEEK(730)
```

Wie bereits erwähnt, kann man auch die Zeitspanne bis die Tastenwiederholfrequenz einsetzt verändern. Dazu dient die Adresse $02D9 bzw. 729. Ihr Inhalt ist bei NTSC auf 48 und bei PAL auf 40 gesetzt. Dieser Wert veranlaßt das Betriebssystem, 0,8 Sekunden zu warten, bis die Tastenwiederholfrequenz einsetzt. Die Zeitspanne errechnet sich wie folgt:

Zeitspanne=(PAL=50/NTSC=60)/PEEK(729)

6. Einstecken und Herausnehmen eines ROM-Modules

Die erhältlichen ROM-Module werden jetzt, genau wie bei den meisten Telespielen, durch die Staubschutzkappe in der Mitte des Gerätes eingesteckt. Bei den alten Geräten mußte man eine Klappe öffnen, um sicherzustellen, daß die ROM's keinen Strom mehr bekommen. Auch gab es bei dem alten ATARI 800 (tm ATARI) die Möglichkeit, ein zweites ROM-Modul einzusetzen. Bei den neuen Geräten braucht man den Computer nicht auszuschalten, wenn man ein ROM-Modul einsetzen oder entfernen will. Das System stellt während des Vertical-Blanks fest, ob eine Veränderung am ROM-Modul stattgefunden hat. Ist dies der Fall, so stellt sich das System durch einen selbst herbeigeführten Kaltstart auf das neue ROM-Modul ein.

7. Selbsttest und BYE-Display

Da das BASIC bei den neuen Geräten eingebaut ist, gibt es keinen MEMO-PAD mehr. Gibt man vom BASIC aus den Befehl BYE ein, so erscheint auf dem Bildschirm das ATARI LOGO mit dem ATARI-Schriftzug. Beim Einschalten des Computers führt dieser selbständig einen Selbsttest durch. Deshalb dauert es bei den neuen Geräten etwas länger, bis die Meldung READY auftaucht. Drückt man die HELP-Taste, während das ATARI LOGO auf dem Bildschirm steht, so kann man in diesen Selbsttest gelangen. Mit den Funktionstasten START, SELECT und OPTION kann man dann verschiedene Funktionen des Computers überprüfen.

8. Zusätzliche Graphikmodi

Die neuen Geräte bieten auch an Möglichkeit neue Graphikstufen durch den GRAPHICS-Befehl aufzurufen, die bei den alten Geräten zwar vorhanden sind, aber nur schwer anzusprechen waren.

GR.	ANTIC-Mode	GITA Mode	Bildspeicherplatz
12	4	0	960
13	5	0	480
14	12	0	3840
15	14	0	7680

Achtung: Werden die neuen Textmodi aufgerufen (12 und 13), so muß man sich einen neuen Zeichensatz definieren, da der eingebaute Zeichensatz falsch entziffert wird. Beachten Sie jedoch, daß eine Zeichenmatrix nicht mehr aus 8*8 Bit besteht, sondern nur noch aus 4*8 Bit. Man darf ein Zeichen also nicht auf einer 8*8 Matrix erstellen, sondern muß eine 4*8 Matrix benutzen. Auf Grund der größeren Farbauswahl wird ein Bit jeweils für die Farbinformationen verwendet.

Aus der Kombination der beiden Bitstellen (die den Matrixpunkt ergeben) bestimmt der Computer die Farbe des Graphikpunktes. Dabei entsprechen folgende Bitkombinationen den aufgelisteten Farbregistern:

00 Farbregister 712
01 Farbregister 708
10 Farbregister 709
11 Farbregister 710 (wenn Bit 7 des Zeichens 0 ist)
11 Farbregister 711 (wenn Bit 7 des Zeichens 1 ist)

Hier noch ein Programm welches anschaulich machen soll, wie ein möglicher Character-Set aussehen kann

```
31000A=INT((INT((PEEK(144)+256*PEEK(145))
/256)+9)/4)*4
31010GRAPHICS 12+16:POKE 87,0:? "  ";
31020POKE 756,A
31030POKE 710,0
31040FOR X=48 TO 57:? CHR$(X);:NEXT X:?
31050FOR X=65 TO 90:? CHR$(X);:NEXT X:?
31060FOR X=0 TO 9:? CHR$(X);:NEXT X:?
31070FOR X=97 TO 122:? CHR$(X);:NEXT X:?
31120FOR X=0 TO 128*8-1
31130READ B
31140POKE A*256+X,B
31150NEXT X
31160END
32000DATA0,0,0,0,0,0,0,0,24,24,24,24,0,
24,0,0,102,102,102,0,0,0,0
32001DATA0,102,255,102,102,255,102,0,24,6
2,96,60,6,124,24,0,0,102,108,24,48,102,70
,0
32002DATA28,54,28,56,111,102,59,0,0,24,24
,24,0,0,0,0,0,14,28,24,24,28,14,0
2,96,60,6,124,24,0,0,102,108,24,48,102,70
,0
32002DATA28,54,28,56,111,102,59,0,0,24,24
,24,0,0,0,0,0,14,28,24,24,28,14,0
32003DATA0,112,56,24,24,56,112,0,0,102,60
,255,60,102,0,0,0,24,24,126,24,24,0,0
32004DATA0,0,0,0,0,24,24,48,0,0,0,126,0,0
,0,0,0,0,0,0,24,24,0
32005DATA0,6,12,24,48,96,64,0,0,4,17,17,1
7,17,4,0,0,4,20,4,4,4,21,0
32006DATA0,21,17,1,4,16,21,0,0,21,1,4,1,1
7,21,0,0,4,4,20,20,21,4,0
32007DATA0,21,16,21,1,17,21,0,0,21,16,21,
17,17,21,0,0,21,1,1,4,4,4,0
32008DATA0,21,17,4,17,17,21,0,0,21,17,17,
21,1,21,0,0,0,24,24,0,24,24,0
32009DATA0,0,24,24,0,24,24,48,6,12,24,48,
24,12,6,0,0,0,126,0,0,126,0,0
32010DATA96,48,24,12,24,48,96,0,0,60,102,
12,24,0,24,0,0,60,102,110,110,96,62,0
```

```
32011DATA0,4,17,17,21,17,17,0,0,21,17,20,
17,17,21,0,0,21,17,16,16,17,21,0
32012DATA0,20,17,17,17,17,20,0,0,21,16,20
,16,16,21,0,0,21,16,20,16,16,16,0
32013DATA0,21,16,16,17,17,21,0,0,17,17,21
,17,17,17,0,0,21,4,4,4,4,21,0
32014DATA0,1,1,1,1,17,21,0,0,17,17,20,20,
17,17,0,0,16,16,16,16,16,21,0
32015DATA0,17,21,21,17,17,17,0,0,17,17,21
,21,17,17,0,0,21,17,17,17,17,21,0
32016DATA0,21,17,17,21,16,16,0,0,21,17,17
,17,21,20,0,0,21,17,17,20,17,17,0
32017DATA0,21,16,21,1,1,21,0,0,21,4,4,4,4
,4,0,0,17,17,17,17,17,21,0
32018DATA0,17,17,17,17,21,4,0,0,17,17,17,
21,21,17,0,0,17,17,4,4,17,17,0
32019DATA0,17,17,21,4,4,4,0,0,21,1,4,4,16
,21,0,0,30,24,24,24,24,30,0
32020DATA0,64,96,48,24,12,6,0,0,120,24,24
,24,24,120,0,0,8,28,54,99,0,0,0
32021DATA0,0,0,0,0,0,255,0,0,8,34,34,34,3
4,8,0,0,8,40,8,8,8,42,0
32022DATA0,42,34,2,8,32,42,0,0,42,2,8,2,3
4,42,0,0,8,8,40,40,42,8,0
32023DATA0,42,32,42,2,34,42,0,0,42,32,42,
34,34,42,0,0,42,2,2,8,8,8,0
32024DATA0,42,34,8,34,34,42,0,0,42,34,34,
42,2,42,0,128,192,224,240,248,252,254,255
32025DATA15,15,15,15,0,0,0,0,240,240,240,
240,0,0,0,0,255,255,0,0,0,0,0,0
32026DATA0,0,0,0,0,0,255,255,0,0,0,0,240,
240,240,240,0,28,28,119,119,8,28,0
32027DATA0,0,0,31,31,24,24,24,0,0,0,255,2
55,0,0,0,24,24,24,255,255,24,24,24
32028DATA0,0,60,126,126,126,60,0,0,0,0,0,
255,255,255,255,192,192,192,192,192,192,1
92,192
32029DATA0,0,0,255,255,24,24,24,24,24,24,
255,255,0,0,0,240,240,240,240,240,240,240
,240
32030DATA24,24,24,31,31,0,0,0,120,96,120,
96,126,24,30,0,0,24,60,126,24,24,24,0
32031DATA0,24,24,24,126,60,24,0,0,24,48,1
26,48,24,0,0,0,24,12,126,12,24,0,0
32032DATA0,24,60,126,126,60,24,0,0,42,34,
34,42,34,34,0,0,42,34,40,34,34,42,0
32033DATA0,42,34,32,32,34,42,0,0,40,34,34
,34,34,40,0,0,42,32,40,32,32,42,0
32034DATA0,42,32,40,32,32,32,0,0,42,32,32
,34,34,42,0,0,34,34,42,34,34,34,0
32035DATA0,42,8,8,8,8,42,0,0,2,2,2,2,34,4
2,0,0,34,34,40,40,34,34,0
32036DATA0,32,32,32,32,32,42,0,0,34,42,42
,34,34,34,0,0,34,34,42,42,34,34,0
32037DATA0,42,34,34,34,34,42,0,0,42,34,34
```

```
,42,32,32,0,0,42,34,34,34,42,40,0
32038DATA0,42,34,34,40,34,34,0,0,42,32,42
,2,2,42,0,0,42,8,8,8,8,8,0
32039DATA0,34,34,34,34,34,42,0,0,34,34,34
,34,42,8,0,0,34,34,34,42,42,34,0
32040DATA0,34,34,8,8,34,34,0,0,34,34,42,8
,8,8,0,0,42,2,8,8,32,42,0
32041DATA0,24,60,126,126,24,60,0,24,24,24
,24,24,24,24,24,0,126,120,124,110,102,6,0
32042DATA8,24,56,120,56,24,8,0,16,24,28,3
0,28,24,16,0
```

9. Änderungen im resistenten Disk Handler

Der resistente Disk Handler, er bootet normalerweise das Disketten Betriebssystem, kann in den alten Geräten nur 128 Bytes pro Sektor verarbeiten. Bei den neuen Geräten ist diese Begrenzung weggefallen. Man kann nun bis zu 65536 Bytes in einen Sektor unterbringen, aber nur dann, wenn der Drive dies auch kann. Die Anzahl der Bytes pro Sektor ist in den Adressen $02D5 (Low Byte) und $02D6 (High Byte) bzw. 725 und 726.

10. Memory-Map

0000 – 4FFF	RAM	00000 – 20479
5000 – 57FF	RAM- oder Selbsttest	20480 – 22527
5800 – 7FFF	RAM	22528 – 32767
8000 – 9FFF	RAM- oder ROM-Modul	32768 – 40959
A000 – BFFF	RAM- oder ROM-Modul	40960 – 49151
C000 – CFFF	OS oder RAM	49152 – 53247
D000 – D0FF	GTIA	53248 – 53503
D100 – D1FF	Reserviert	53504 – 53759
D200 – D2FF	POKEY	53760 – 54015
D300 – D3FF	PIA	54016 – 54271
D400 – D4FF	ANTIC	54272 – 54527
D500 – D5FF	Jeder Zugriff auf eine dieser Adressen übergibt die ROM-Modul-Kontrolle an das ROM-Modul Interface	54528 – 54783
D600 – D7FF	Reserviert	54784 – 55295
D800 – FFFF	OS oder RAM	55296 – 65535

11. Fine Scrolling

Die neuen Geräte bieten die Möglichkeit, die DISPLAYLIST auf vertikales Fine Scrolling umzuschalten. Dies mußte bisher sehr mühsam per Hand erledigt werden. Um das Fine Scrolling einzuschalten, muß die Adresse $026E ein $FF bzw. die Adresse 622 eine 255 enthalten. Danach muß man einen Graphikmod, in dem das Fine Scrolling gebraucht wird, neu öffnen. Nun ist die DISPLAYLIST auf vertikales Fine Scrolling eingestellt und kann, wie sonst üblich, angesprochen werden. Soll das Fine Scrolling abgeschaltet werden, so ist der Wert der Adresse $026E auf $00 zu ändern. Die Schritte sind noch einmal einzeln aufgeführt.

Einschalten des Fine Scrolling:

1. POKE622,255
2. OPEN#1,12,0,"E"
3. PROGRAMM FÜR FINE SCROLLING
4. CLOSE #

Abschalten des Fine Scrolling:

1. POKE622,0
2. OPEN#1,12,0,"E"
3. CLOSE#

Und nun noch ein kleines Programm für vertikales Fine Scrolling:

```
 10   VSCROL=54277
 30   POKE 622,255
 40   OPEN #1,12,0,"E"
 50   LTST
 60   FOR X=0 TO 7
 70   POKE VSCROL,X
 80   GOSUB 100
 90   NEXT X:END
100   FOR I=0 TO 50:NEXT I:Return
```

12. Voice Box

Und nun noch ein kurzes Wort zu der eingebauten Voice Box der Serie 1400. Um die eingebaute Voice Box anzusprechen, muß man einen Kanal öffnen.

```
OPEN#1,4,0"V1:PF"
```

Um nun ein Wort durch den Computer sprechen zu lassen, muß man das Wort in eine phonetische Buchstabenfolge umsetzen. Hat man die phonetische Folge in einem String abgelegt, so braucht man nur noch folgenden Befehl einzugeben:

```
PRINT #,STRING
```

Ein Programm für den ATARI 1450 XL sieht dann wie folgt aus:

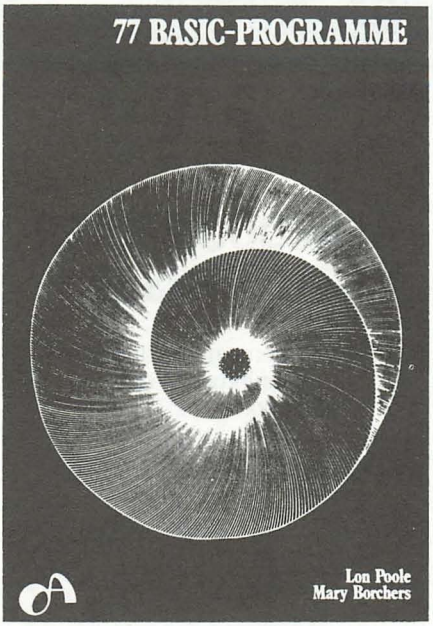
```
10 DIM A$(10)
20 A$="HELLO"
30 OPEN#1,4,0,"V1:PF"
40 PRINT#1;A$
```

Die Programmierung ist also sehr einfach.

Viel Spaß nun bei der Programmierung der neuen ATARI-Computer wünschen Ihnen

Die Autoren

77 BASIC–PROGRAMME



Dieses Buch enthält eine Sammlung von praktischen 77 Kurzprogrammen, die in der populären Programmiersprache BASIC geschrieben sind. Es werden mathematische, finanztechnische, statistische und verschiedene allgemeine Aufgaben behandelt.
Die leichtverständlich erläuterten Beispiele können direkt und selbst ohne jede Erfahrung in BASIC verwendet werden.
Jedes Programm wird zunächst kurz beschrieben und später mit Hinweisen auf Änderungsmöglichkeiten der entsprechenden Programme aufgeführt.
Die Programme haben einen Umfang von etwa 15 bis 130 Zeilen BASIC.
Das Buch entstand unter Verwendung eines CBM-Computers und ist somit eine besonders nützliche Hilfe für jeden Besitzer dieses beliebten Computers.
Jedoch lassen sich die Programme auch ohne große Änderungen und bei einiger Sorgfalt auf anderen bekannten Systemen verwenden.
Der Ausdruck erfolgte auf einem Heathkit-Drucker WH-14. Bei der Verwendung anderer Drucker mit unterschiedlicher Zeilenlänge können die Print-Anweisungen innerhalb der Programme entsprechend geändert werden.
Die Darstellung der Programme ist ebenso auf einem Bildschirm möglich.

77 BASIC-Programme – Ein Buch, das Spaß am Nutzen bringt.
– von Lon Poole und Mary Borchers
208 Seiten, Softcover, DM 39,–

---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

MIKROCOMPUTER-GRUNDWISSEN

Eine allgemeinverständliche Einführung in die Mikrocomputer-Technik – vom Mikrocomputer-Papst Dr. Adam Osborne. Optimal als Einstieg für Elektronik-Laien; zur Kontaktaufnahme mit diesem die Technik und unsere Umwelt revolutionierenden Gebiet. Dieses Buch ist aber kein trockener Lernstoff, sondern in seiner didaktisch ausgezeichneten und humorvollen Form – mit vielen Illustrationen und Cartoons – ein Werk, das man verschlingt. Und dabei spielend leicht ein Thema begreift, das unser Leben immer mehr beherrscht.

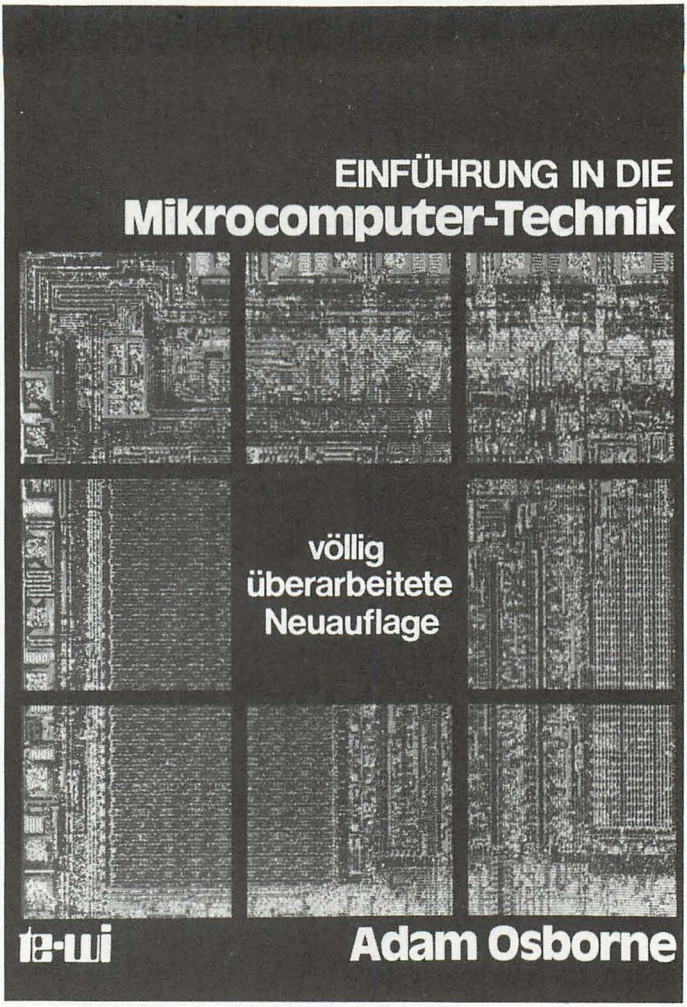
MIKROCOMPUTER-GRUNDWISSEN – von Adam Osborne
304 Seiten, Paperback, DM 36,–

---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40



te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

EINFÜHRUNG IN DIE MIKROCOMPUTER-TECHNIK

Der Titel ist fast schon legendär – das Buch jetzt aber völlig neu. Osborne's bestens bekanntes Standardwerk, dieser absolute Bestseller, der auch an den Universitäten weltweit eingeführt ist, hat eine drastische Überarbeitung erfahren, die den allerletzten Stand der Technik wiederspiegelt. Geblieben ist der bewährte Stil des Autors mit seiner ungewöhnlichen Fähigkeit, eine der bedeutsamsten Technologien unserer Zeit in Bildern statt nur in Worten erklären zu können. Beim Durchlesen dieses Buches wird kaum ein Gedanke zu finden sein, der nicht als Bild anschaulich gemacht wurde. Diese Einführung in Prinzipien, Funktion und Systemaufbau von Mikrocomputern vermittelt eine lebendige und umfassende Vorstellung vom Wesen dieser Technik udn befähigt den Leser, konkrete Aufgaben durch richtige Fragen an Experten oder durch Verständnis der Systembeschreibung von Mikrocomputern zu lösen. Man lernt nicht nur, die Komponenten eines Mikrocomputersystems richtig zu beurteilen und für die Eignung auf die anvisierten Zwecke auswählen zu können, sondern erhält auch Antwort auf die Fragen der Programmierung, Vergleiche alternativer Mikrocomputersysteme, der Wirtschaftlichkeit bei Anwendung in Massenprodukten und erfährt den Sachverstand zur Beurteilung zukünftiger Entwicklungen.

Einführung in die Mikrocomputer-Technik – Das bekannte Standardwerk; jetzt völlig neu überarbeitet
– von Adam Osborne
480 Seiten, Hardcover, DM 66,–

---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

6502 – PROGRAMMIEREN IN ASSEMBLER

Wenn Sie den Mikroprozessor 6502 selbst in der Assemblersprache programmieren wollen, ist dies das richtige Buch für Sie. Dieser Band, der eine ganze Serie von deutschsprachigen Büchern über die Programmierung in Assembler repräsentiert, enthält über 80 praktische Programmierbeispiele im Standardformat einschließlich Flußdiagrammen, Quellprogrammen, Objektcodes und erläuternden Texten.
Jeder Befehl des Mikroprozessors 6502 wird detailliert erklärt.

Weitere Themen u.a.:

- Assembler-Vereinbarungen des 6502
- Programmierung des Unterbrechungsprogramms des 6502
- E/A-Bausteine und Interfacing-Verfahren des 6502

Ausführliche Besprechungen für die Erstellung von Programmen, von der Definition der Aufgabe, über Testen, Fehlersuche, Dokumentation, bis hin zu modularer und strukturierter Programmierung runden dieses aktuelle Werk ab.
Die Darstellung ist außerordentlich gründlich und didaktisch gut vorbereitet.

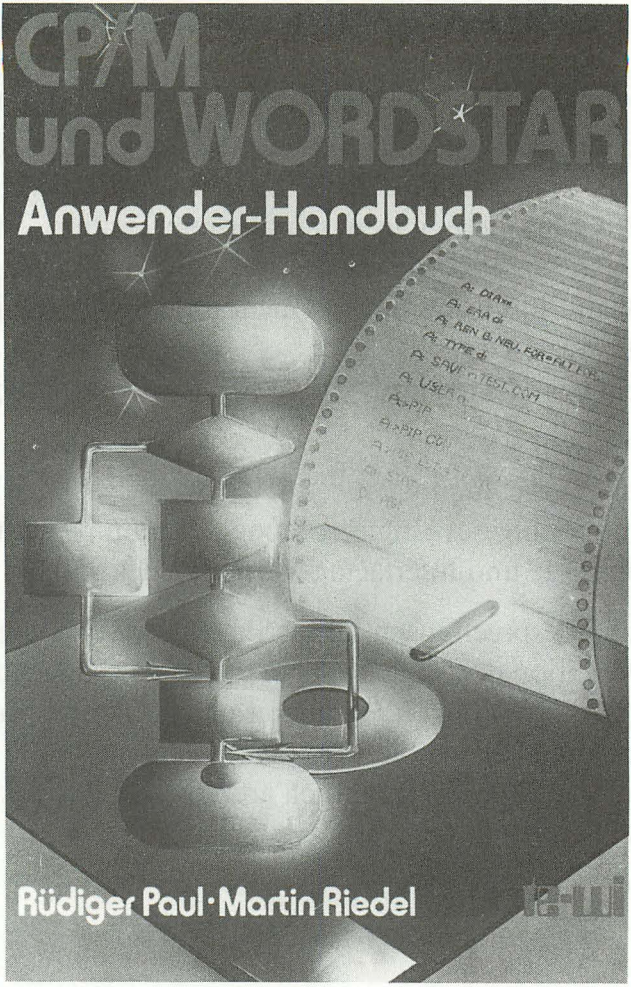
6502 – PROGRAMMIEREN IN ASSEMBLER
Ein unentbehrliches Hilfsmittel für den Hardware-Praktiker, das auch dem Einsteiger ein leichtes Einarbeiten bietet.
– von Lance A. Leventhal
700 Seiten, Paperback, DM 59,–

---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40



te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

CP/M und WORDSTAR

Orientiert man sich im Computerbereich an Systemen amerikanischer Herkunft, die die Anwender-Zielgruppe unterer und teilweise mittlerer Bereiche betrifft, stößt man immer wieder auf CP/M.
Für dieses Software-System zum Betrieb von Computern ist nun ein Standardwerk in deutscher Sprache erschienen, das dem ständig steigenden Kreis von Mikrocomputer-Benutzern eine fundamentale Einarbeitungshilfe bietet. Dieses Buch baut auf den ursprünglichen Unterlagen über CP/M von Digital Research einerseits auf, setzt diese jedoch nicht voraus. Es werden die grundsätzlichen Zusammenhänge des BIOS beschrieben, wie auch das Mehrbenutzer-Betriebssystem MP/M, das auf CP/M aufbaut. Als logische Vervollständigung rundet die Beschreibung über das Textsystem "WordStar" den gebotenen Inhalt sinnvoll ab.
Für alle, die das Riesen-Reservoire an durchweg kommerzieller Software beanspruchen und darüberhinaus selbst Programme entwickeln wollen.

CP/M und WORDSTAR – von Rüdiger Paul und Martin Riedel
144 Seiten, Paperback, DM 29,80

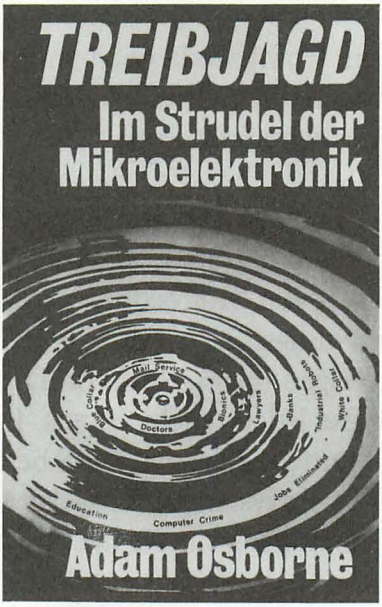
---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

TREIBJAGD – IM STRUDEL DER MIKROELEKTRONIK



Bekommen wir die durch die Mikroelektronik eingeleitete industrielle Revolution in den Griff? Oder sie uns –
In diesem Taschenbuch erläutert Osborne die Ursprünge für die gigantische Entwicklung in der Mikrocomputer-Industrie, deckt Hintergründe auf, nennt erschreckende Fakten, warnt uns eindringlich vor den gesellschaftlichen Problemen, die diese Revolution auslösen wird, zeigt aber auch Wege, wie wir sie in den Griff bekommen können.
Populär und packend geschrieben, erhalten Sie Insider-Information aus erster Hand – vom Mikrocomputer-Papst aus Silicon-Valley.

TREIBJAGD – Im Strudel der Mikroelektronik – von Adam Osborne
208 Seiten, Paperback, DM 36,–

---

te-wi Verlag GmbH
Telefon 089/1 29 20 90

te-wi

Theo-Prosel-Weg 1
8000 München 40

COMPUTER FÜR KINDER

Ein Buch für Kinder und ihre Lehrer – ein kindgemäßes Buch für die erste Begegnung mit Computern, ihren Eigenwilligkeiten, und ihren unerschöpflichen Möglichkeiten.
Ein Buch zu unserer Gegenwart und zur Zukunft unserer Kinder.
„Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern keines der unzähligen Computer-Bücher geschrieben wurde.
„Computer für Kinder" ist ganz auf Kinder eingestellt und beschäftigt sich unterhaltsam und leicht verständlich mit folgenden Themen:

Wie arbeiten Computer
Wie funktioniert mein Computer
Wie programmiert man mit einfachen Flußdiagrammen
Wie kann ich BASIC leicht verstehen
Programme aufbauen mit Befehlen
Farbige Graphiken entwerfen
Erklärung von Computer-Begriffen

Sally Greenwood Larson war Kindergärtnerin, ehe sie selbst Computern begegnete und zwischen den Welten von Kindern und Computern zu vermitteln begann.
Computer für Kinder, A4 quer, Fadenheftung, über 100 Seiten, DM 29,80

te-wi

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

LOGO

Jeder kann programmieren
Computersprache für Eltern und Kinder
DANIEL WATT

LOGO ... Ergebnis der Erforschung menschlicher Intelligenz

Entwickelt von Seymour Papert, Pädagoge und Mathematikprofessor.
Erste Computersprache, die bewußt Strategien menschlichen Denkens dient – und in ihrer Logik der Realität gerecht wird. LOGO ersetzt BASIC, sagen Pädagogen und Mathematiker. LOGO kommt dem übergreifenden, assoziativen Denken entgegen. BASIC dagegen ist ein Setzkasten von Logik-Buchstaben.

DANIEL WATT ... hat im Team von Seymour Papert gearbeitet und ein Buch geschrieben, das voller Bilder seine Erlebnisse mit Kindern am Computer wiedergibt. Ein hochwertiges Textbuch für LOGO-Kurse. Ein Buch für Eltern die mit ihren Kindern nicht "Computer", sondern "Lust am eigenen Denken" erleben wollen.

ca. 400 Seiten, DM 59,–

"Buch des Jahres 1983" in den USA

te-wi

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Wer mit Computern zu tun hat, weiß, daß das Wissen über den Computer der Schlüssel zur Nutzung seiner Fähigkeiten ist.
Dieses Buch macht die Möglichkeiten, die in Ihrem ATARI®-Computer stecken, auf leichtverständliche Art transparent.
In einfachen Schritten wird der Anwender mit der Bedienung der Geräte und der Software vertraut gemacht.
Zahlreiche Tips zur Aufdeckung und Beseitigung von möglichen Fehlerquellen bei Hard- und Software helfen bei scheinbar unlösbaren Problemen.
In zwei Kapiteln erlernen Sie die einzigartigen Grafikmöglichkeiten Ihres ATARI®-Computers spielerisch einzusetzen.
Ein in sich abgeschlossener ATARI-BASIC-Kurs macht schnell mit dieser einfachen Programmiersprache vertraut.
Der Umgang mit Farbe, Grafik und Ton wird Ihnen anhand von Text und Programmen vermittelt.
Dem fortgeschritteneren Programmierer bieten die vielen Übersichtstafeln, sowie die alphabetische Auflistung der BASIC-Befehle und Funktionen, eine entscheidende Hilfestellung.

Weitere te-wi Bücher

Computer für Kinder
(Sally Greenwood Larson)
Ein Buch für Kinder, ihre Lehrer und Eltern. „Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern dieses Buch bewußt geschrieben wurde. Unterhaltsam und leicht verständlich.

NEU! LOGO Computersprache für Kinder und Eltern (Daniel Watt)
Dieses Buch beweist: **Jeder kann programmieren.** LOGO ist die Computersprache für Eltern und Kinder. Nicht umsonst wurde dieser Titel zum „Buch des Jahres 1983" in den USA. LOGO ist das Ergebnis der Erforschung menschlicher Intelligenz; entwickelt von einem Pädagogen und Mathematikprofessor. LOGO ist die erste Computersprache, die bewußt Strategien menschlichen Denkens dient.

Sprühende Ideen mit ATARI®-Grafik
(Tom Rowley)
Ein Lehrbuch, das mit den Grafikmöglichkeiten des ATARI-Computers in die Gestaltgesetze von Objekten, in Farbgebung und die Entwicklung von Bildschirmentwürfen einführt.

Mikrocomputer-Grundwissen
(A. Osborne)
Eine allgemeinverständliche Einführung in die Mikrocomputer-Technik – optimal als Einstieg für Elektronik-Laien.

6502 – Programmieren in Assembler
(Lance A. Leventhal)
Ein Titel über die Assemblersprachen-Programmierung. Er ist für den Hardware-Praktiker ein unentbehrliches Hilfsmittel und ermöglicht dem Einsteiger ein leichtes Einarbeiten in die Assemblersprache.

CP/M und WordStar Anwender-Handbuch
(Rüdiger Paul/Martin Riedel)
Für das CP/M-Software-System ist dieses Standardwerk in deutscher Sprache erschienen, das dem ständig steigenden Kreis von Mikrocomputer-Benutzern eine fundamentale Einarbeitungshilfe bietet.

77 BASIC-Programmme
(Lon Poole/Mary Borchers)
Diese Buch beschreibt 77 Kurzprogramme, die finanztechnische, mathematische, statistische und verschiedene allgemeine Aufgaben mit Programmbeispielen behandeln.

ISBN 3-921803-18-7